BAYESIAN DATA ASSOCIATION FOR TEMPORAL SCENE UNDERSTANDING

by

Ernesto Brau Avila

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF COMPUTER SCIENCE

In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY

In the Graduate College
THE UNIVERSITY OF ARIZONA

2013
As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Ernesto Brau Avila entitled Bayesian data association for temporal scene understanding and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Date: 13 December 2013

Kobus Barnard

Date: 13 December 2013

Alon Efrat

Date: 13 December 2013

Clayton Morrison

Final approval and acceptance of this dissertation is contingent upon the candidate’s submission of the final copies of the dissertation to the Graduate College. I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Date: 13 December 2013

Dissertation Director: Kobus Barnard
STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Ernesto Brau Avila
ACKNOWLEDGEMENTS

I want to express my gratitude to everyone who contributed to the work presented in this dissertation. To the members of the IVILAB, a big thanks for providing a stimulating work environment, and for their interesting discussions and comments on several aspects of this work. I especially want to thank Jinyan Guan for her enormous contribution, without which this would not have been possible. Many thanks also to Kyle Simek, Luca Del Pero, Colin Dawson, and Andrew Predoehl, who directly contributed to this project. I also want to thank undergraduate students Phillip Lee and Zewei Jiang, who wrote software and collected and annotated data crucial to the success of this work.

I am extremely grateful to my supervisor Dr. Kobus Barnard, for providing ongoing encouragement, support, and, perhaps most of all, patience. His mentorship was crucial in my development as a researcher and an academic. Many thanks to the other members of my committee, Dr. Clayton Morrison and Dr. Alon Efrat. I am also very thankful to Dr. Ravi Palanivelu for giving me the opportunity to contribute to the project which eventually led to the work presented here. I would also like to acknowledge the financial support I obtained from CONACyT, the National Science Foundation, and the Defense Advanced Research Projects Agency, which allowed me to focus on my research over the course of my studies. Also, thanks to the Department of Computer Science and to the School of Information: Science, Technology, and Arts for providing me with teaching opportunities. Very special thanks to Rhonda Leiva, Cheryl Craddock, Holly Brown, and Bridget Radcliff for their pivotal administrative support.

Finally, I want to thank my family for their support and solidarity. Thanks to my parents for their continuing love and encouragement, and to my brothers Agustín and Julio for their friendship. I am especially grateful to my wife Jazmín and to my son Ernesto for providing a loving home environment, and for helping me and putting up with me at times when I needed it the most. Without you none of this would have been possible.
DEDICATION

To my parents, Agustín and Guadalupe, for bestowing upon me an insatiable thirst for knowledge.

To my brothers, Agustín and Julio, who showed me the true meaning of camaraderie.

To my wife Jazmín and my son Ernesto, for being an infinite source of inspiration and motivation.
TABLE OF CONTENTS

LIST OF FIGURES ......................................................... 8

LIST OF TABLES .......................................................... 10

ABSTRACT ................................................................. 11

CHAPTER 1 INTRODUCTION ............................................... 13
  1.1 Related work ....................................................... 18
    1.1.1 Multiple target tracking .................................... 18
    1.1.2 Human pose estimation ....................................... 23
    1.1.3 Statistical inference ......................................... 24

CHAPTER 2 BAYESIAN DATA ASSOCIATION FOR TEMPORAL SCENE
  UNDERSTANDING ......................................................... 26
  2.1 Introduction ....................................................... 26
  2.2 Model ............................................................... 27
    2.2.1 Data association .............................................. 28
    2.2.2 Scene and imaging system .................................... 31
    2.2.3 Likelihood .................................................... 33
  2.3 Inference .......................................................... 34
    2.3.1 MCMC sampling for scene understanding ..................... 35
    2.3.2 Estimating the scene ......................................... 42
  2.4 Evaluation ........................................................ 42
  2.5 Discussion ......................................................... 44

CHAPTER 3 TRACKING MULTIPLE SMOOTH TRAJECTORIES .............. 46
  3.1 Introduction ....................................................... 46
  3.1.1 Related work ................................................... 49
  3.2 Model ............................................................... 50
    3.2.1 Scene and microscope ........................................ 50
    3.2.2 Likelihood .................................................... 51
  3.3 Inference .......................................................... 51
    3.3.1 Gibbs data association ...................................... 52
  3.4 Data, tuning, and ground truth .................................. 53
  3.5 Results ............................................................. 56
    3.5.1 Evaluation .................................................... 57
TABLE OF CONTENTS – Continued

3.5.2 Gaussian process as smooth motion .......................... 58
3.5.3 Inference .......................................................... 60
3.6 Discussion ............................................................ 61

CHAPTER 4 TRACKING PEOPLE IN 3D ................................. 63
4.1 Introduction ........................................................... 63
4.1.1 Related work ...................................................... 64
4.2 Model ................................................................. 65
4.2.1 Scene and camera ............................................... 65
4.2.2 Likelihood ......................................................... 68
4.3 Inference ............................................................. 72
4.4 Data, tuning, and ground truth ................................. 77
4.5 Results ............................................................... 80
4.5.1 Experiments ....................................................... 81
4.6 Discussion ........................................................... 82

CHAPTER 5 TRACKING PEOPLE’S GAZE IN 3D ...................... 85
5.1 Introduction ........................................................... 85
5.1.1 Related work ...................................................... 86
5.2 Model ................................................................. 87
5.2.1 Scene and camera ............................................... 87
5.2.2 Likelihood ......................................................... 91
5.3 Inference ............................................................. 93
5.4 Data, tuning, and ground truth ................................. 94
5.5 Results ............................................................... 95
5.6 Discussion ........................................................... 96

APPENDIX A IMPLEMENTATION DETAILS ......................... 98
A.1 Software ............................................................ 98
A.2 Efficient gradient computation ................................. 98

APPENDIX B MCMC SAMPLING ........................................ 102
B.1 Background ........................................................ 102
B.2 Metropolis-Hastings ............................................. 102
B.3 Gibbs ............................................................... 104
B.4 Hamiltonian Monte Carlo .................................... 104

REFERENCES .......................................................... 107
LIST OF FIGURES

1.1 Example video frame .............................................. 14
1.2 Example video frame .............................................. 15
1.3 Visual summary of model ........................................... 16

2.1 Example association ................................................ 28
2.2 Association as a partition: example ............................... 32
2.3 Graphical model of general joint distribution ...................... 34
2.4 General sampler ..................................................... 36
2.5 Association sampling moves ........................................ 39
2.6 Grow track forward .................................................. 40
2.7 Procedure to grow track forward ................................... 41
2.8 CLEAR metrics ....................................................... 43

3.1 Sample pollen tube images .......................................... 48
3.2 Gibbs data association ............................................... 54
3.3 Pollen tube detection process ....................................... 55
3.4 Tracked pollen tubes ................................................. 60
3.5 Tracked birds ......................................................... 61

4.1 Model box ............................................................. 67
4.2 Occlusion reasoning .................................................. 68
4.3 Likelihood function .................................................. 71
4.4 Association sampling moves ........................................ 73
4.5 Grow track forward .................................................. 74
4.6 Example person detections .......................................... 78
4.7 Example optical flow output ......................................... 79
4.8 Qualitative results .................................................... 83

5.1 Person model .......................................................... 88
5.2 Model boxes .......................................................... 90
5.3 Optical flow likelihood ............................................... 92
5.4 Face landmark likelihood ............................................. 93
5.5 Example landmark detections ....................................... 94
5.6 Qualitative results .................................................... 97

B.1 The MH algorithm ................................................... 104
B.2 The Gibbs algorithm .................................................. 105
LIST OF FIGURES – Continued

B.3 The HMC algorithm ................................................. 106
LIST OF TABLES

<table>
<thead>
<tr>
<th></th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Results for tracking pollen tubes</td>
<td>59</td>
</tr>
<tr>
<td>3.2</td>
<td>Results: GP as smooth motion</td>
<td>59</td>
</tr>
<tr>
<td>3.3</td>
<td>Results: inference method</td>
<td>61</td>
</tr>
<tr>
<td>4.1</td>
<td>Benchmark results</td>
<td>80</td>
</tr>
<tr>
<td>4.2</td>
<td>Calibration results</td>
<td>82</td>
</tr>
<tr>
<td>4.3</td>
<td>Impact results</td>
<td>82</td>
</tr>
<tr>
<td>5.1</td>
<td>Benchmark results</td>
<td>96</td>
</tr>
<tr>
<td>5.2</td>
<td>Head orientation results</td>
<td>96</td>
</tr>
</tbody>
</table>
ABSTRACT

Understanding the content of a video sequence is not a particularly difficult problem for humans. We can easily identify objects, such as people, and track their position and pose within the 3D world. A computer system that could understand the world through videos would be extremely beneficial in applications such as surveillance, robotics, biology. Despite significant advances in areas like tracking and, more recently, 3D static scene understanding, such a vision system does not yet exist. In this work, I present progress on this problem, restricted to videos of objects that move in smoothly and which are relatively easily detected, such as people. Our goal is to identify all the moving objects in the scene and track their their physical state (e.g., their 3D position or pose) in the world throughout the video.

We develop a Bayesian generative model of a temporal scene, where we separately model data association, the 3D scene and imaging system, and the likelihood function. Under this model, the video data is the result of capturing the scene with the imaging system, and noisily detecting video features. This formulation is very general, and can be used to model a wide variety of scenarios, including videos of people walking, and time-lapse images of pollen tubes growing in vitro. Importantly, we model the scene in world coordinates and units, as opposed to pixels, allowing us to reason about the world in a natural way, e.g., explaining occlusion and perspective distortion. We use Gaussian processes to model motion, and propose that it is a general and effective way to characterize smooth, but otherwise arbitrary, trajectories.

We perform inference using MCMC sampling, where we fit our model of the temporal scene to data extracted from the videos. We address the problem of variable dimensionality by estimating data association and integrating out all scene variables. Our experiments show our approach is competitive, producing results
which are comparable to state-of-the-art methods.
CHAPTER 1

INTRODUCTION

People can understand the content of a video sequence without much effort. Indeed, we can easily identify and track the moving objects in a video, understand the way they interact with each other, and even reason about the 3D layout of the scene from the flat image frames. For example, consider the video frame in Figure 1.1. It is easy to recognize that there are three people in the image, two of whom are walking together. We also can see that two of them are looking at each other. We are not confused when two of them become occluded by the other, as in the later frame shown in Figure 1.2. Further, we are able to estimate distances and other measurements, such as height and velocity, in absolute coordinates.

This kind of understanding of the world from videos plays a key role in many higher-level cognitive processes. For instance, the people standing close to each other and looking at each other might indicate a conversation is taking place. Similarly, a good estimate of the 3D heights of people provides cues as to their age.

Despite significant advances over the last decades, there is no computer vision system which can achieve this level of understanding of these kinds of scenes. Indeed, there has been much progress in solving individual tasks like object recognition, visual tracking, activity recognition, and many others; however, a vision system with human-like capabilities remains elusive. To be precise, we do not yet have a system which is able to identify and track all moving objects of interest in a scene, as well as estimate their physical state, all in absolute 3D units and coordinates. Additionally, such a system would have to address the fact that the camera which produced the video is often unknown.

In this dissertation we present our work in temporal scene understanding, applied to videos containing moving objects with two constraints, The objects must be relatively easy to detect; e.g., people detectors have become extremely effective in the
Figure 1.1: Understanding a video is simple for humans. In this frame, we can easily identify that there are three people, two of whom are walking together and looking at each other. A third person is standing in place holding a box. We wish to explore the idea of jointly inferring the 3D geometry of the scene and the camera, as opposed to a 2D tracking of the image pixels. Given such a video, we want to jointly recover the 3D position and some physical state of any relevant moving objects at every frame, as well as the camera parameters.

last few years. In addition, the objects must exhibit relatively smooth motion, where we define a smooth trajectory as one where correlation between positions varies exponentially with temporal distance. We wish to provide a global 3D representation of the scene featured in such videos, using prior information about the objects’ geometry, their motion, and the camera. More specifically, given a video, our goal is to jointly infer: (1) the camera parameters; (2) the number of moving objects of interest in the scene, which is unknown; (3) the 3D position of all objects at every frame; and (4) the relevant physical state of all objects at every frame. Further, we want to estimate the last two quantities in world coordinates and units.

We propose to do this using a Bayesian generative approach, where we separately model the data association, the scene, the camera, and the video creation process. More specifically, we assume that the video features were statistically generated by
Figure 1.2: Although one person is completely occluded by another one, we have no trouble following their path. A 3D representation of the scene makes very natural, and allows us to understand occlusion at the model level, as opposed to handling it during inference. In other words, occlusion provides information, rather than confusion.

projecting the 3D scene onto the image plane at every frame of the video. This Bayesian framework is very powerful, and has the benefit of separating modeling and inference. Further, we separately model the association of object detections and the 3D scene and camera, allowing us to make representation decisions in a principled way during the modeling phase. During inference, we search the space of scenes by drawing samples from the posterior distribution, which has two components, the prior distribution and the likelihood function. The first enforces any prior knowledge we have about the scene, such as the motion of objects or their physical dimensions. We use the likelihood to compare a hypothesized scene with the video data by projecting it with the hypothesized camera. Figure 1.3 summarizes our model visually.

One of the main contributions of this work is the Bayesian generative model, which considers (explicitly) the data association, the 3D temporal scene, and the
camera. Each of these components is modeled separately and individually, resulting in a very general model which is applicable to many different scenarios. The model for data association is based on well-known work by Oh et al. (2004), but we extend it by allowing multiple observations to be assigned to each object. Modeling data association explicitly (as opposed to estimating it as a side-effect of inference) has several benefits. First, it allows control over its representation at the model level, making it simpler to modify and extend. Additionally, it provides a way to enumerate models, essentially giving us a principled way to perform model choice.
A second key aspect of our approach is the fully-3D representation of the temporal scene. Specifically, we use 3D primitives, such as points, cylinders, and ellipsoids, to represent moving objects, and use a Gaussian processes to model their motion. The camera is represented separately, but its specific model depends on the video being processed. Having this global understanding of the 3D scene and camera helps address ambiguities arising from the imaging process, such as clutter, occlusion, and perspective distortion. For example, rigid objects in 3D do not change as they move in space, but may appear very different in different frames of the video. Similarly, objects may become occluded for a portion of the video; under a 3D representation this is a source of evidence, whereas it may create confusion in a 2D view. In addition, our representation allows us to incorporate prior knowledge about objects, a fact which we exploit, for example, by imposing a general smoothness prior over motion, implemented via a Gaussian process.

Finally, we exploit the Bayesian generative model and the 3D representation to explore the space of solutions efficiently. The core of our inference approach is Markov chain Monte Carlo (MCMC) sampling, which is very useful for problems of high-dimensionality. We use the independence properties of our probabilistic model to our advantage, and recursively sample over the different components of the model, in such a way that current hypotheses for one informs the inference for the others. Additionally, our 3D representation lets us safely avoid impossible configurations, such as two objects occupying the same space, which we prohibit during inference. Finally, since we explicitly model the data association variables, we are able to perform model choice in a principled way. More specifically, we do this by integrating out scene variables, which do not have constant dimensionality. In other words, we attempt to find the association and camera pair which has the best marginal posterior by performing a weighted average of all possible 3D configurations given that data association and camera.

In the final section of this chapter, we discuss the problem background and examine any relevant previous work. The rest of the dissertation is organized as follows. In Chapter 2, we introduce our approach to temporal scene understanding
in a general context. In Chapter 3, we describe the result of using our approach on videos of pollen tubes growing in a petri dish. In Chapter 4, we discuss the application of our approach to videos of scenes featuring people walking. Finally, in Chapter 5, we use a more sophisticated model of people which includes head orientation.

1.1 Related work

In this section, we review related work, which, for clarity, we separate it into three categories. First, we discuss the literature on multiple target tracking (Section 1.1.1). We then briefly consider previous work on human pose estimation (Section 1.1.2). Finally, in Section 1.1.3, we examine any relevant work on statistical inference.

1.1.1 Multiple target tracking

Target tracking is a very important task in computer vision. In particular, the problem of tracking multiple objects has recently gained notoriety, in large part due to the proliferation of high-powered computers, the wide availability of inexpensive, high quality video cameras, and the increasing need for automatic video analysis. Indeed, multiple target tracking has important applications in tasks such as

- surveillance – automated monitoring a scene and identifying suspicious activity
- traffic monitoring – real-time collection of traffic statistics in order to direct traffic
- vehicle navigation – automatically plan paths while avoiding obstacles
- human-computer interaction – e.g., head tracking and gesture recognition in virtual reality environments

In its simplest form, we can define multiple target tracking (MTT) as the problem of estimating the trajectories of objects (targets) as they move about a scene. To do this, a tracker must assign labels to objects consistently across different frames
of the video. Additionally, trackers may also estimate other object information, such as velocity, pose, and shape. MTT can be a difficult problem, and there are several factors which contribute many common challenges. First, the projection of a 3D world onto a 2D image plane loses information, and causes artifacts such as occlusion, perspective distortion and foreshortening, and object overlap and crossing. Additionally, camera quality and preprocessing causes noise in the frames and in any extracted features. Finally, the large variety of types of objects and the interactions between them result in a wide range of complex motions.

In this review, we will limit ourselves to the problem of tracking-by-detection, since it is most closely related to our work. Let us start with perhaps the most widely-used tracking method in the literature, the Kalman filter first introduced by Kalman et al. (1960), where it was shown that, if motion is assumed to be linear-Gaussian, and noise is also Gaussian, the exact trajectory for a single target can be computed exactly in linear time. This model has been used extensively in tracking over the past few decades. Broida and Chellappa (1986) used the Kalman filter to track points in noisy images. Beymer and Konolige (1999) used it in the context of stereo camera-based people tracking.

Although very useful and simple, the Kalman filter suffers from one the disadvantage, which is that the motion model is much too naive. There are many examples of motion which are not linear, and where noise is not Gaussian (we tackle this type of problem in this dissertation). However, once this constrain is relaxed, the solution to the estimation problem no longer has closed form. One way people have worked around this is by using particle filters, which keep an estimate of the object trajectory represented by samples from the posterior. This method was introduced by Arulampalam et al. (2002), and is also known as sequential Monte Carlo, because it estimates the posterior with a set of samples. Variations of the particle filter have been used in tracking with success, such as in work by Isard and Blake (1998), Okuma et al. (2004), and Yang et al. (2005).
Data association for MTT

Let us turn now to the most important challenge faced when solving the multiple target tracking problem: data association (DA). In this typical scenario, multiple measurements are observed in each frame of a video, but it is not known a priori which measurement comes from which object. There are many approaches to this problem, and they vary significantly. On the probabilistic side, one of the classical methods for solving the DA problem is the joint probabilistic data association (JPDA) filter, reviewed in Bar-Shalom et al. (2009), which attempts to use a weighted sum of probabilities to estimate the correct association. JPDA has been used extensively, such as in work by Chang and Aggarwal (1991), where they use it to solve the structure from motion problem. Later, Rasmussen and Hager (2001) used a constrained JPDA to track regions. The major limitation of JPDA is its inability to track new objects entering the scene, since it assumes a fixed number of objects throughout the video.

A method which does not suffer from this drawback is the multiple hypothesis tracker (MHT) (Reid, 1979), which keeps several hypotheses for the correct assignment, delaying the choice of association until enough frames have been observed. Cox and Hingorani (1996) presented an efficient implementation of the MHT algorithm which tracked multiple targets with some success. Cham and Rehg (1999) used a multiple hypothesis approach to track the entire human body.

Finally, Oh et al. (2004) introduced their now-well-known approach for data association, which they named MCMC for data association (MCMCDA). In this work, they present a set of sampling moves to sample from the posterior distribution of associations given detections. Further, they showed that MCMCDA theoretically reaches the optimal filtering solution to the data association problem. Yu and Medioni (2009) used this approach for MTT, where they allowed spatial correction of detections. More recently, Brau et al. (2011) applied an extended version of MCMCDA to track multiple smooth trajectories.

Other MCMC data association techniques have been proposed. One which is
closely related to MCMCDA is the work of Benfold and Reid (2011), in which they use MCMC sampling moves to explore the space of associations. Their proposals are quite different than the standard MCMCDA moves, however. Another technique based on particle filters was introduced by Khan et al. (2005), where they use MCMC sampling to determine the association of measurements to targets, and they use particle filters to estimate the trajectory.

More recently, several deterministic approaches to data association have been contributed. For example, Berclaz et al. (2009) cast the problem as a flow problem, and use an integer linear program to solve it. Andriyenko and Schindler (2010) further extend this approach by discretizing the space of target locations. Wu et al. (2011) also use a network flow approach, but they use a complicated linking scheme where they associate nodes in a graph together based on the flow. In later work, Andriyenko et al. (2012) pose data association as a multi-labeling problem, and minimize an energy function associated to a discrete Markov random field.

**People tracking**

Tracking people in videos is an area which has seen much attention in the last decade or so. In the typical setup, we have a video of a common scene – such as a street, a park, or a hallway – and there are multiple people moving freely (usually walking) about it. There are many approaches to this problem; the following is a very brief summary of the recent work.

Isard and MacCormick (2001) proposed a method which is largely related to the work presented here. They track people in 3D using 3D primitives to represent each person. More specifically, they use a generalized 3D cylinder as their person model. The main difference in their approach is perhaps that they are not using the tracking-by-detection paradigm; they instead use a filtering approach, where they consider all pixels as their measurements, which is an good way to circumvent the data association problem. Practical differences between this and our work include their constraint on the number of people in the scene, and their need for a pre-calibrated camera.
The last three years have seen a surge in the number of people tracking papers. In 2010, Andriluka et al. (2010) a three-stage approach to tracking and pose estimation: build a person detector, extract tracklets from small number of consecutive frames (producing more detections), use evidence accumulated to recover 3D pose. Andriyenko and Schindler (2010) posed the problem as a network flow problem, and solved it using an integer linear program. Choi and Savarese (2010) model interaction between targets explicitly, in addition to estimating certain camera parameters, and infer their parameters using a particle filtering approach.

Andriyenko and Schindler (2011) proposed a similar approach to ours, in that they explore the space of 3D (ground plane) trajectories, using an energy-minimization scheme with trans-dimensional jumps. However, they do not address the model choice problem. The same group later proposed a way to reason about occlusion in an anlytical way (Andriyenko et al., 2011), in which they represent the image of each person with a 2D Gaussian, making the overlap computation very efficient.

The work of Benfold and Reid (2011) also bears some similarity with our work. They an MCMC approach to solve the data association problem, including moves which move detections across targets. They also perform Metropolis-Hastings steps to estimate the trajectories. However, their representation is quite different, and they do not address model choice. Mohedano and Garcia (2011) also use MCMC to perform 3D people tracking and camera estimation. Similarly, they do not address model choice. Additionally, data association is not modeled explicitly.

Andriyenko et al. (2012), like us, attempt to solve both the association and trajectory problem simultaneously. However, they use restrict motion to be linear-Gaussian, and do not deal with the variable-dimensional space in a principled way. Yan et al. (2012) take a very different approach, where they have multiple trackers and detectors at each frame, and choose the best result from among them. Roshan Zamir et al. (2012) use a graph-theoretic approach, where they pose data association as a minimum clique problem, yielding excellent results. Wu et al. (2012) express the detection and multiple target tracking problems in a single, global ob-
jective function, in an attempt to avoid problems caused by ineffective detectors.

1.1.2 Human pose estimation

Pose estimation has received a significant amount of attention in the last few years, perhaps due in part to the recent widespread availability of computing power. The problem of pose estimation is that of estimating the pose of one or more people in an image or video. Evidently, this is a form of image understanding, and it is related to our work.

One way to estimate human pose is by using spatial models, which encode the structure of the human body. For instance, ensemble of parts approaches attempt to detect different body parts in consistent configurations. Felzenszwalb (2011) uses a grammar model for 2D pose estimation and object detection (Felzenszwalb et al., 2009). In this work, compositional rules are used to represent objects as combinations of others. Ensemble approaches can also be used in 3D, such as in work by Sigal et al. (2004), where a skeletal model was used in conjunction with multi-view data.

In the context of tracking, Sidenbladh et al. (2000) and Choo and Fleet (2001) have done full 3D body estimation from 2D video using MCMC methods, using a 3D skeletal model of the human body, and using particle filters and Hybrid Monte Carlo to estimate the parameters of their Bayesian model from data. This method is highly related to our work, but there are several key differences. First, they track the pose of a single person, whereas we consider multiple targets. Additionally, they assume the camera is known a priori and that the initial pose is given.

Head pose estimation

Head pose estimation has also received some attention in the past few years, and there are several approaches to it. Appearance template methods compare a new image of a head to a set of exemplars in order to find the most similar view. This method was used by Niyogi and Freeman (1996) and Beymer (1994) with some
success. Another approach is to use nonlinear regression methods, such as those used by Huang et al. (1998) and Jones and Viola (2003).

In the context of tracking, one can estimate head pose by tracking features, such as Zhao et al. (2007), who used RANSAC to register head features across frames. Yang and Zhang (2002) also tracks feature, but uses binocular video. More similar to our work are those who do model-based tracking, such as Wu and Toyama (2000), La Cascia et al. (2000), and Xiao et al. (2003), all of whom attempt to fit a 3D model to tracked features across a video.

More recently, there has been work applying particle filters with a non-linear regression approach to the problem, as in work done by Murphy-Chutorian and Trivedi (2008). Also, the problem has been posed as a graph matching problem. Such work includes that of Wu and Trivedi (2008), who uses Kernel Latent Data Association to solve the association problem.

1.1.3 Statistical inference

Given the dimensionality of our model, and the non-linearities introduced by the camera transformation, we cannot perform inference analytically. Instead, we resort to approximate inference, relying on Markov chain Monte Carlo (MCMC) methods (Neal, 1993) to sample from the posterior distribution.

MCMC methods are often used to solve integration and optimization problems in cases when exact inference is not an option. These techniques are used to draw samples from distributions which cannot be simulated directly, but that can be evaluated up to a constant factor. MCMC works by building a Markov chain which as the target distribution as its invariant distribution; upon simulating this chain, we generate samples from the target distribution. For a general overview of MCMC methods for probabilistic inference problems, we refer to the work of Neal (1993), and to the work of Andrieu et al. (2003). Further, Forsyth et al. (2001) reviews MCMC approaches to Bayesian inference problem in the domain of computer vision.

In this work, we combine several different MCMC algorithms to explore the vast parameter space. We use a Gibbs approach to sample over the different compo-
nts of our model. The Gibbs sampler works by successively sampling from the full conditional posterior distributions. Although it possesses desirable theoretical qualities, it is difficult to apply in practice, since the full-conditionals are usually difficult to sample from. In our case, we use different MCMC algorithms to sample from these distributions. We provide a very brief summary of the techniques we use in Appendix B.

To sample the space of associations, we use the Metropolis-Hastings algorithm, which requires a proposal distribution which is easy to sample. We extend the work of Oh et al. (2004), who first proposed a set of general association sampling moves. We note that this approach falls under the category of data driven MCMC, first proposed by Zhu et al. (2000), which formalizes the idea of conditioning on the data when sampling from the proposal distribution. We use the Hamiltonian Monte Carlo (HMC) algorithm (Neal, 2011) to draw samples from the posterior over camera parameters, following Del Pero et al. (2011) and Schlecht and Barnard (2009), who used it successfully to sample models with a similar parametrization. We also use HMC to sample over the space of 3D scenes. In particular, we sample over the space of object trajectories, which bears some similarity to the work of Choo and Fleet (2001) and the work Sidenbladh et al. (2000).
2.1 Introduction

In this chapter, we introduce our approach to the problem of temporal scene understanding, which we define thusly. Consider a group of objects (called targets) moving continuously in a scene for a finite period of time. At every time point, a target’s state is made up of its position, and any other property of interest (e.g., size or pose), and is specified in world units and with respect to the world coordinate system. At discrete times (frames) we observe the noisy state of the targets via an imaging system (e.g., a camera), as well as false observations produced by a noise process. These observations are called detections and are measured in image coordinates. For example, when tracking people, the scene is the 3D world, the imaging system is a perspective camera, and the detections are the output of an object detector.

The objective is to estimate the number of targets and their state at every frame, given the set of detections. We are also interested in estimating the data association, which specifies which detections were produced by which targets. Here we discuss our approach in a general setting; in subsequent chapters, we apply the ideas developed here to the specific tracking tasks of tracking pollen tubes in vitro (Section 3), tracking people in 3D (Section 4), and tracking gaze direction in 3D (Section 5).

We begin by discussing a Bayesian generative formulation (Section 2.2), in which we separately model data association, the scene and imaging system, and the process by which they generate our data, which consists of detections and any other image features. Modeling data association explicitly is a key aspect of our approach, and
provides a principled way to separate discrete and continuous variables, as well as simplifying inference by allowing us to avoid model choice problems (Section 2.2.1).

Our scene model (Section 2.2.2) consists of the state of all targets at each frame. Each target has a position in $\mathbb{R}^D$, and can have other state variables, all of which are modeled in world units. Having a scene representation in world coordinates has several advantages. We can naturally incorporate prior information, e.g., motion models are in meters as opposed to pixels. Also, such a representation allows reasoning about the scene, e.g., occlusion provides evidence rather than confusion. We also model the imaging system, which, along with the scene and noise model, generates the detections and any other image data of interest (Section 2.2.3).

As mentioned above, the ultimate goal of temporal scene understanding is to estimate the targets’ states. However, the variable dimensionality of the space of scenes could make it difficult to navigate. Similarly, we encounter the problem of model choice, since scenes of different dimensions are not directly comparable. To avoid this, we marginalize out the scene variables, and maximize the posterior over associations (given the data). In other words, we find a MAP estimate of the association which generated the data (Section 2.3).

2.2 Model

We now discuss our generative model in greater detail. Let $B_t$ be the set of detections obtained in frame $t$, and $B = \cup_{t=1}^{T} B_t$, where $T$ is the length of the video. We start by generating an association $\omega$ from the prior $p(\omega)$, which encodes the number of targets $m$, their start and end frame, and the number of detections they produce. We then generate the scene $z$, which consists of the states of all targets at every frame, from the prior $p(z \mid \omega)$, and the imaging system $C$ from $p(C)$. Finally, the set of detections $B$ gets generated from the detection likelihood $p(B \mid z, C, \omega)$, and any other image data $I$ from the image likelihood $p(I \mid z, C, \omega)$. The remainder of this section explores each of these components in detail. See Figure 2.3 for the graphical representation of the model.
2.2.1 Data association

An association is an object which contains information about tracks: their first frame, their length, and the number of detections they produced at each frame, as well the number of false detections at every frame. An example association is shown in Figure 2.1. The association entity is based on the that described in well-known work by Oh et al. (2004), but we extend that work by (1) allowing tracks to produce multiple measurements at any given frame and (2) employing a prior on associations which allows parameters governing track dynamics and detector behavior to adapt to the environment of a particular video See Figure 2.3a for the graphical model.

<table>
<thead>
<tr>
<th>( t )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_t )</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( d_t )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( n_{t} )</td>
<td>( \bullet )</td>
<td>( \bullet )</td>
<td>( \bullet )</td>
<td>( \bullet )</td>
</tr>
<tr>
<td>( a_{rt} )</td>
<td>( \bullet )</td>
<td>( \bullet )</td>
<td>( \bullet )</td>
<td>( \bullet )</td>
</tr>
</tbody>
</table>

Figure 2.1: Graphical depiction of a sample from the generative model for associations. In the table, \( t \) is the frame, and \( e_t \) and \( d_t \) are the number of targets entering and exiting the scene at frame \( t \). \( n_t \) and \( a_{rt} \) show the detections generated by noise and by target \( r \), respectively, each of which is represented by a different-colored dot. For example, in frame \( t = 2 \), two targets entered the scene, no targets exited it, there was one false alarm (shown in black), target \( \tau_1 \) produced zero detections (shown in blue), and targets \( \tau_2 \) and \( \tau_3 \) (resp. red and green) produced one and two detections.
Tracks

Let $m_1$ be the number of objects present when the movie begins, and $e = (e_t)_{t=2}^T$ be the number of objects entering the scene at each frame with $e = \sum_{t=2}^T e_t$ the total number of entrances, and $m = m_1 + e$ the total number of objects in the movie. Let $l = (l_r)_{r=1}^m$ be the observed length of each track (the true amount of time the object spends in the scene is potentially latent, as it may enter or leave before outside the temporal extent of the video) with $l = \sum_{r=1}^m l_r$. Finally, let $m_t$ be the number of objects present at the $t^{th}$ frame and $d_t$ the number of objects exiting after the $t^{th}$ frame, and let $m = (m_t)_{t=1}^T$, $d = (d_t)_{t=1}^{T-1}$, and $d = \sum_{t=1}^{T-1} d_t$. The set of tracks is uniquely specified by $(m_1, e, l)$; however we introduce the extra notation for interpretability of formulae.

We model tracks as an infinite server queue with Markovian arrivals and Markovian service (the $M/M/\infty$ queue (Hoel et al., 1972)), which has attained its stationary distribution at the start of the movie. That is, we suppose that (1) the intervals between entrance times, and the intervals between entrance and exit times, are all mutually independent, that (2) there is no upper bound on the number of objects that can be present at one time, and that (3) there is no prior significance attached to the starting or ending times of the movie. Then

$$e_t \iid \sim \text{Pois} (\nu)$$  \hspace{1cm} (2.1)

$$l'_r \iid \sim \text{Exp} (\theta),$$  \hspace{1cm} (2.2)

where $l'_r$ becomes $l_r$ after censoring at the end of the movie. It can be shown that, under these assumptions, the stationary distribution of $m_1$ is $\text{Pois}(\nu/\theta)$. Letting $\kappa = \nu/\theta$, we get (assuming stationarity from the outset)

$$p(m_1, e, l | \kappa, \theta) = p(m_1 | \kappa) p(e | \kappa, \theta) p(l | m_1, e, \theta)$$  \hspace{1cm} (2.3)

$$= \frac{e^{-\kappa} \kappa^{m_1}}{m_1!} \left[ \prod_{t=2}^T e^{\kappa \theta (e^t)} \frac{e^t}{e_t!} \right] \left[ \prod_{r=1}^d \theta e^{-\theta l_r} \right] \left[ \prod_{r=d+1}^M e^{-\theta l_r} \right],$$  \hspace{1cm} (2.4)

where the tracks are indexed according to when they exit the scene, and the last
term reflects the censoring of the \(l_r\). Eq. 2.4 can be re-factored as

\[
p(m_1, e, l | \kappa, \theta) = \frac{\kappa^{m_1} e^{d} e^{-(\kappa+(T-1)\kappa\theta+\theta)}}{m_1! \prod_{t=2}^{T} e_t!}.
\] (2.5)

Since the number of tracks and their lengths are properties of the video, we put vague Gamma priors on \(\kappa\) and \(\theta\):

\[
p(\kappa) \propto \kappa^{\alpha_\kappa-1} e^{-\beta_\kappa \kappa}
\] (2.6)

\[
p(\theta) \propto \theta^{\alpha_\theta-1} e^{-\beta_\theta \theta}
\] (2.7)

We sample the values of \(\kappa\) and \(\theta\) during inference (see Section 2.3).

**Detects and correspondence**

In frame \(t\), each of \(m_t\) tracks and a noise process produces detections. Let \(n = (n_t)_{t=1}^{T}\) be the number of detections due to noise in each frame, with \(n = \sum_{t=1}^{T} n_t\), and let \(A = ((a_{it})_{i=1}^{m_t})_{t=1}^{T}\), with \(a_t = \sum_{i=1}^{m_t} a_{it}\) and \(a = \sum_{t=1}^{T} a_t\). (There is a slight abuse of notation in the first index of \(a_{it}\): the \(i\)th track in frame \(t\) is not necessarily the track labeled \(i\) throughout the whole video.) We also define \(N_t = n_t + a_t\) as the total number of detections at time \(t\). Note that \(|B_t| = N_t\). We suppose that

\[
n_t \overset{i.i.d.}{\sim} \text{Pois}(\lambda_N)
\] (2.8)

\[
a_{it} \overset{i.i.d.}{\sim} \text{Pois}(\lambda_A),
\] (2.9)

where \(\lambda_A\) is specified beforehand and and \(\lambda_N\) is given the vague Gamma prior

\[
p(\lambda_N) \propto \lambda_N^{\alpha_N-1} e^{-\beta_N \lambda_N},
\] (2.10)

and its value is inferred during the sampling process. We justify this by observing that the value of \(\lambda_A\) is mostly a property of the detectors being used, whereas \(\lambda_N\) is a property of the video being studied. The prior for \((n, A)\) is then

\[
p(n, A | m_1, e, l, \lambda_N) = p(n | \lambda_N)p(A | m_1, e, l)
\] (2.11)

\[
= \prod_{t=1}^{T} \frac{e^{-\lambda_N} \lambda_N^{n_t}}{n_t!} \prod_{i=1}^{m_t} \frac{e^{-\lambda_A} \lambda_A^{a_{it}}}{a_{it}!}.
\] (2.12)
Finally, we suppose that a fully-specified assignment in a frame is a permutation of the $N_t$ detections (both true and noise) at time $t$, with the first $n_t$ associated to noise, the next $a_{1t}$ associated to the first track in the frame, etc. We model per-frame associations as full permutations of the detections (rather than just a partition) in order to hold constant the number of possible assignments given the data and the number of tracks. This is important in order to be able to compare posterior probabilities on the same scale during inference. Putting all this together and letting $e_1 = m_1$, we have the following prior over associations:

$$p(\omega | \kappa, \theta, \lambda_N, \lambda_A) = \frac{p(n, A | m_1, e, l, \lambda_N) p(m_1, e, l | \kappa, \theta)}{\prod_{t=1}^{T} N_t!}$$

$$= \frac{(\kappa e^{-\lambda_A})^{m_1} \theta e^{+d} \lambda_N^N \lambda_A^a \exp(-\kappa + (T - 1) \kappa \theta + l \theta + T \lambda_N)}{\prod_{t=1}^{T} (N_t! e_t! n_t! \prod_{i=1}^{m_t} a_{it}!)}.$$

(2.13)

(2.14)

where $\omega$ is called a data association, and which has two different interpretations.

First, a data association is made up of $m_1$, $e$, $l$, $n$, and $A$), and tells us which detections (once they are generated) will correspond to each target. Similarly, we can view $\omega$ as a partition of the set of detections $B$; more specifically, we let $\tau_r \subset B$, $r = 1, \ldots, m$ be the set of detections associated to target $r$; $\tau_0$ be the set of false alarms, where $|\tau_0| = n$. Then, we simply let $\omega = \{\tau_0, \ldots, \tau_m\}$. We call the $\tau_r$, $r \neq 0$ tracks, and $\tau_0$ the noise track. We shall denote the $j$th detection of track $\tau_r$ with $\tau_{rj}$, where we assume detections are ordered by frame. Additionally, we define the function $t_i(\tau)$, which gives us the frame of $\tau$’s first detection; similarly, $t_f(\tau)$ gives us the frame of $\tau$’s final detection. Finally, we also denote $\gamma = (\kappa, \theta, \lambda_N)$ and group the hyperparameters $\phi_\omega = (\alpha_\kappa, \beta_\kappa, \alpha_\lambda, \beta_\lambda, \alpha_\theta, \beta_\theta)$, so that

$$p(\gamma | \phi_\omega) = p(\kappa | \alpha_\kappa, \beta_\kappa) p(\theta | \alpha_\theta, \beta_\theta) p(\lambda_N | \alpha_\lambda, \beta_\lambda).$$

(2.15)

2.2.2 Scene and imaging system

The scene $z = (z_1, \ldots, z_m)$ consists of the states of all targets. Each target’s state is a trajectory $x_r$ and whichever other quantities are of interest $w_r$, so that $z_r = $
Figure 2.2: An example of the association-as-partition view. Here, detections are simply points on the image plane, represented by the circles. The number inside each circle is the frame in which it was detected, and the color represents the track, with the unfilled circles represent $\tau_0$ (set of false detections). (a) shows an empty association, i.e., every detection is in $\tau_0$, while (b) illustrates an association with two tracks, made up of the blue and red points, respectively.

$(x_r, w_r)$. We represent trajectories as a sequence of points world coordinates $x_r = (x_{r1}, \ldots, x_{rl_r})$, with $x_{rj} \in \mathbb{R}^{D_x}$, for all $r = 1, \ldots, m$ and $j = 1, \ldots, l_r$. Note that the second index of each $x_{rj}$ is not a frame, but the $j$th point in target $r$’s trajectory. To obtain the frame corresponding to a particular trajectory point $x_{rj}$, we simply take the starting frame of target $r$ and add $j$ frames (i.e., $t_i(\tau_r) + j$). However, we shall often abuse notation and use $x_{rt}$, implicitly assuming that target $r$ enters the scene at frame 1, without loss of generality. It is also worth noting that $x_r$ is specified for every $j = 1, \ldots, l_r$, even for components corresponding to frames in which it generated no detections. In other words, target $r$ still exists in the scene even in frames where it goes undetected.

The prior on $x_r$ comes from motion, which we model using Gaussian processes (GP). More specifically, we model each dimension of the trajectory separately as an independent realization of a GP with zero mean and squared-exponential covariance function with inputs $S_r = \{1, \ldots, l_r\}$. In other words, if we let $x^i_r$ be the vector of the $i$th components of the elements of $x_r$, then, for $i = 1, \ldots, D_x$, we have that $x^i_r \sim \mathcal{N}(0, K_r)$, where $K_r$ is a $l_r$-by-$l_r$ matrix whose element $(s, s')$ is given by the covariance function $k(s, s') = \sigma^2_x \exp(-\frac{1}{2\nu_x}(s - s')^2$ for all pairs in $S_r \times S_r$. The
signal variance and scale parameters $\sigma_x^2$ and $l_x$ are set beforehand. We further assume that the trajectory dimensions are independent of one another, so that $p(x_r) = \prod_{i=1}^{D_x} p(x_i^r)$.

The prior over the state of target $r$ is given by $p(z_r) = p(x_r)p(w_r | x_r)$, where the prior over $w_r$ depends on the specific type of targets being tracked. (In the chapters that follow, we discuss specific state variables for different applications of the model.) Finally, we assume targets to be independent of one another, giving us a scene prior of

$$p(z | \phi_z, \phi_w) = \prod_{r=1}^{m} p(z_r | \phi_z, \phi_w),$$

(2.16)

where $\phi_z = (\sigma_x, l_x)$ and $\phi_w$ are any parameters to $p(w_r | x_r)$. Finally, we have a prior on the imaging system $p(C | \phi_C)$, which is also domain-dependent, and has distributional parameters $\phi_C$.

### 2.2.3 Likelihood

The final stage of the generative model is the likelihood function $p(B, I | \omega, z, C)$ which is what governs the generation of the detections $B$ and other image data $I$. Here, we discuss the general characteristics of the likelihood function; for specific likelihoods on different tracking problems, see chapters 3, 4, and 5.

We assume that $B$ and $I$ are conditionally independent, and that the distribution in above can be factored into two different likelihoods, the detection likelihood $p(B | \omega, z, C)$ and the image likelihood $p(I | \omega, z, C)$. Further, we assume the individual detections in $B$ to be conditionally independent given the target $r$ which generated them and the frame $t$ in which they exist, so that

$$p(B | \omega, z, C) = \prod_{b \in B} p(b | z_{rt}, C),$$

(2.17)

where $z_{rt} = (x_{rt}, w_{rt})$ is the full state of target $r$ at frame $t$. For the time being, we shall not make any assumptions about the exact forms of the detection and image likelihoods, and simply stipulate that they have distributional parameters $\phi_B$ and
\[ p(B, I \mid \omega, z, C, \phi_B, \phi_I) = p(B \mid \omega, z, C, \phi_B)p(I \mid \omega, z, C, \phi_I). \] (2.18)

2.3 Inference

The generative model described in Section 2.2 gives us a joint distribution of the form

\[ p(\omega, \gamma, z, C, B, I \mid \phi) = p(\omega \mid \gamma)p(\gamma)p(z)p(C)p(B, I \mid \omega, z, C), \] (2.19)

where the factors are given by Eqs. 2.14, 2.15, 2.16, and 2.18, and where we have omitted the dependence on parameters \( \lambda_A, \phi_\omega, \phi_x, \phi_w, \phi_C, \phi_B, \) and \( \phi_I \) on the right.
handside for clarity. Since our goal is to find a good value for the scene \( z \), we might consider maximizing the posterior distribution \( p(\omega, \gamma, z, C \mid B, I) \). However, this approach suffers from several disadvantages related to the variable dimensionality of the space of scenes. For instance, one must be careful in designing jump moves which efficiently move between subspaces of different dimensionality. Even more concerning is the problem of model choice, which, as Green (1995) showed, is very difficult to handle.

To circumvent this, we opt for marginalizing out \( z \) altogether, and maximizing the posterior

\[
p(\omega, \gamma, C \mid B, I) = p(\omega \mid \gamma)p(\gamma)p(C)p(B, I \mid \omega, \gamma, C) \tag{2.20}
\]

\[
= p(\omega \mid \gamma)p(\gamma)p(C) \int_z p(B, I \mid \omega, z, C)p(z) \, dz. \tag{2.21}
\]

Since \( \omega, \gamma, \) and \( C \) are all of constant dimensionality, we do not encounter the problems discussed above. However, this approach poses its own difficulties, in that the integral in Eq. 2.21 could be intractable in some domains. We face this challenge when tracking people in 3D, which is discussed Chapter 4. For now, we shall assume that the integral can be computed (or approximated), and focus our attention on our approach to maximizing Eq. 2.20.

We use Markov chain Monte Carlo (MCMC) sampling to maximize the posterior (Eq. 2.20) with respect to \( \omega, \gamma, \) and \( C \). MCMC is a family of techniques used to generate samples from a probability distribution \( \pi(\cdot) \), called the target distribution. More specifically, the idea is to build a Markov chain that has \( \pi \) as its invariant distribution; then, simulating chain produces samples from \( \pi \), which, in our case, is the posterior (Eq. 2.20). Specifically, we use the Gibbs and Metropolis-Hastings algorithms for our purposes. We provide some background and more details of these and other MCMC algorithms in Section B.

2.3.1 MCMC sampling for scene understanding

We use both the MH and Gibbs algorithms to draw samples from the posterior distribution. At the highest level, we sample from the conditional posteriors
Algorithm 2.4: Gibbs for sampling \((\omega, \gamma, C)\)

**Input:** Initial state \((w^{(0)}, C^{(0)}, \gamma^{(0)})\)

for \(i = 1, \ldots, N\) do

**Sample \(\gamma\):**

Draw \(\kappa^{(i)} \sim p(\kappa | \theta^{(i-1)}, \omega^{(i-1)})\) \(\triangleright\) (eq. 2.25)

Draw \(\theta^{(i)} \sim p(\theta | \kappa^{(i)}, \omega^{(i-1)})\) \(\triangleright\) (eq. 2.26)

Draw \(\lambda^{(i)}_N \sim p(\lambda_N | \omega^{(i-1)})\) \(\triangleright\) (eq. 2.27)

**Sample \(C\):**

Draw \(C^{(i)} \sim p(C | \omega^{(i-1)}, \gamma^{(i)}, B, I)\) \(\triangleright\) (problem-specific)

**Sample \(\omega\):**

Draw move \(v \sim q_V(\cdot | \omega^{(i-1)})\)

Propose \(\omega' \sim q_v(\omega | \omega^{(i-1)})\)

Let \(p_{\text{accept}}\) be the acceptance probability

\[
\max \left( 1, \frac{p(\omega' | \gamma^{(i)}, C^{(i)}, B, I) q(\omega^{(i-1)} | \omega')}{p(\omega^{(i-1)} | \gamma^{(i)}, C^{(i)}, B, I) q(\omega' | \omega^{(i-1)})} \right)
\]

Draw \(u \sim U(0,1)\)

if \(u < p_{\text{accept}}\) then

Set \(\omega^{(i)} = \omega'\)

else

Set \(\omega^{(i)} = \omega^{(i-1)}\)


Figure 2.4: Gibbs algorithm for sampling from the joint posterior over \(\omega, \gamma,\) and \(C\). See Section 2.3.1 for details.

To sample \(\gamma = (\kappa, \theta, \lambda_N)\), we again use Gibbs and sample from the conditional posteriors. Here, we take advantage of the conditional independence in our model,
and note that

\begin{align}
p(\kappa \mid \theta, \lambda, \omega, C, D) &= p(\kappa \mid \theta, \omega) \\
p(\theta \mid \kappa, \lambda, \omega, C, D) &= p(\theta \mid \kappa, \omega) \\
p(\lambda_N \mid \kappa, \theta, \omega, C, D) &= p(\lambda_N \mid \omega). 
\end{align}

Then, using conjugacy of the hyper-priors on \(\kappa, \theta, \) and \(\lambda_N\) (Eqs. 2.6, 2.7, 2.10), we get the conditional posteriors

\begin{align}
\kappa \mid \theta, \omega &\sim G(m + \alpha_{\kappa}, 1 + (T - 1)\theta + \beta_{\kappa}) \\
\theta \mid \kappa, \omega &\sim G(e + d + \alpha_{\theta}, l + (T - 1)\kappa + \beta_{\theta}) \\
\lambda_N \mid \omega &\sim G(n + \alpha_{\lambda}, T + \beta_{\lambda}),
\end{align}

from which it is easy to obtain samples.

Sampling from the conditional posterior on \(C\) is application-specific, and will not be discussed here. See subsequent chapters for examples.

**MCMC for data association**

We use the Metropolis-Hastings algorithm to generate samples from the conditional posterior \(p(\omega \mid \gamma, C, B, I)\). We develop a proposal mechanism based on well-known work by Oh et al. (2004), which consists of several sampling moves that construct associations by creating or modifying tracks from the current association. More precisely, we have seven ways (moves) to propose an association: (1) birth, (2) death, (3) extension, (4) reduction, (5) merge, (6) split, and (7) switch, whose proposal distributions we denote by \(q_1, \ldots, q_7\), respectively. At each iteration, given the current association \(\omega = \{\tau_0, \ldots, \tau_m\}\), we draw \(v \sim q_V(\cdot \mid \omega)\), which is a distribution over the set of moves \(\{1, \ldots, 7\}\), usually set to \(U(1, \ldots, 7)\). (There are two special cases here worth noting: when \(|\omega| = 0\), then \(q_V(1) = 1\) and \(q_V(v) = 0\) for \(v = 2, \ldots, 7\); when \(|\omega| = 1\), then \(q_V(5) = 0\) and \(q_V(7) = 0\), since we need at least two tracks to perform merge and switch moves.) Then, using the chosen move, we propose a new association \(\omega' \sim q_v(\cdot \mid \omega)\), where \(\omega' = \{\tau_0', \ldots, \tau_m'\}\). The proposal
probability is then given by $q_\omega(\omega' | \omega) = q_V(v | \omega)q_v(\omega' | \omega)$ and the reverse proposal probability by $q_\omega(\omega | \omega') = q_V(v' | \omega')q_v(\omega | \omega')$, where $v'$ is the reverse move of $v$. See Figure 2.4 for details about this procedure.

We now discuss the sampling moves in detail. Note that this approach is data-driven, which means that the proposal distribution depends on the data $B$ (in fact, we should denote the proposal distribution $q_\omega(\cdot | \cdot, B)$, but omit the dependence on $B$ for readability). Specifically, we shall assume that each detection $b \in B$ has a position in the image, and we will simply use $b$ to denote it. Finally, as before, let $\omega = \{\tau_0, \ldots, \tau_m\}$ be the current association sample. See Figure 2.5 for an illustration of the seven moves.

**Birth move** We select a frame $t_0$ uniformly at random from $\{1, \ldots, T\}$. and choose a detection $b_0$ also u.a.r. from $\tau_0 \cap B_{t_0}$, i.e., the set of false alarms at frame $t_0$, and initialize the new track $\tau'$ to $\{b_0\}$. We then grow $\tau'$, either forward or backward with probability $\frac{1}{2}$ by successively adding detections to it using the following procedure (without loss of generality, assume forward growth). Let $t$ be the current end frame for $\tau_0$. For frame $t' = t + 1$, we fit a line to the detection locations of the previous $s$ frames and let $b'$ be the predicted point on the line at frame $t'$. If $\tau'$ is fewer than $s$ frames long, $b'$ is set to the last detection at frame $t$. We then independently choose detections from the available set $\tau_0 \cap B_{t'}$ at random with probability proportional to their squared distance from $b'$, and add them to $\tau'$. If none of the detections from frame $t'$ were assigned, we stop growing with probability $c$; otherwise, we let $t = t'$ and repeat. Figure 2.6 illustrates the growth procedure, and Figure 2.7 has more details about it. The resulting track $\tau'$ is then added to the current association to get $\omega' = \{\tau'_0, \tau_1, \ldots, \tau_m, \tau'\}$, where $\tau'_0 = \tau_0 \setminus \tau'$ is the updated noise track.

**Death move** To kill a track, we choose $r$ u.a.r. from $\{1, \ldots, m\}$, and let $\omega' = \omega \setminus \{\tau_r\}$, where we also replace $\tau_0$ with $\tau_0 \cup \tau_r$. The birth move and death move are reverse moves of each other.
Figure 2.5: Illustration of the seven MCMC proposal moves we use. Each figure contains an example a move, with the arrow indicating the direction of the move (i.e., $\omega \rightarrow \omega'$). Detections are represented as points on the image plane, and the numbers inside indicate in which frame they exist. White circles represent false alarms, and different colored circles represent different tracks (red and blue, in this case).
Extension move  We choose a track $\tau_r$ u.a.r. from $\omega$. We then grow it forward or backward to produce $\tau'_r$ using the procedure described above (Figure 2.6). The resulting association is $\omega' = (\omega \setminus \{\tau_r\}) \cup \{\tau'_r\}$, where, as before, the noise track of $\omega'$ is given by $\tau_0 \setminus (\tau'_r \setminus \tau_r)$, which is the old noise track minus the detections added to $\tau_r$. is the reduction move.

Reduction move  We choose a track $\tau_r$ u.a.r. from $\omega$. and pick a detection $\tau_{rj}$ u.a.r. from $\{\tau_{r2}, \ldots, \tau_{rL-1}\}$, i.e., any detection of $\tau_r$ except the first and last one. We then remove all detections from the track after or before $\tau_{rj}$ to get, for example, $\tau'_r = \{\tau_{r1}, \ldots, \tau_{rj}\}$ for a reduction from the end. The resulting association is $\omega' = (\omega \setminus \{\tau_r\}) \cup \{\tau'_r\}$, whose noise track is $\tau_0 \cup (\tau_r \setminus \tau'_r)$. The reduction and extension moves are reverse moves of each other.

Merge move  We assign a weight to each pair of tracks $(\tau_{r1}, \tau_{r2})$ proportional to $q_1(\tilde{\omega} | \omega)$, where $\tilde{\omega}$ is an association which only contains one track, $\tau_{r1} \cup \tau_{r2}$, and the appropriate noise track. In other words, the weight represents the probability of creating track $\tau_{r1} \cup \tau_{r2}$ with a birth move. We then normalize the weights and choose a pair of tracks under those probabilities. The resulting track is $\tau' = \tau_{r1} \cup \tau_{r2}$. 

Figure 2.6: Illustration of the procedure to grow a track forward. The red dots indicate $S$, the set of detections for the last $s$ frames of the growing track $\tau$, and the black dots are the set of available detections at frame $t$, $\tau_0 \cap B_t$, which are all candidates to be appended to $\tau$. The red line is fit to the detections, and each detection in $\tau_0 \cap B_t$ is added to $\tau$ with probability given by the Gaussian surface placed around the predicted point on the line, represented by the concentric circles on the diagram. See Section 2.3.1 and 2.7 for more details.
Algorithm 2.7: Grow track forward

**Input:** track \( \tau = \{\tau_1, \ldots, \tau_{|\tau|}\} \), past detection window \( s \)

Let \( t = t_f(\tau) + 1 \)

while \( t \leq T \) do

\( t = t_f(\tau) + 1 \) \( \triangleright (\tau \text{ is longer than } s \text{ frames}) \)

if \( t_f(\tau) - t_i(\tau) + 1 \geq s \) then

Let \( S \) be the set of detections corresponding to \( \tau \)’s last \( s \) frames

Let \( l_S \) be a line fit through points in \( S \)

Let \( b' = l_S(t_f(\tau)) \)

else

Let \( b' = \tau_{|\tau|} \)

for all \( b \in \tau_0 \cup B_{|\tau|} \) do

Let \( p \propto -\|b - b'\|^2 \)

Append \( b \) to \( \tau \) with probability \( p \)

if appended 0 detections to \( \tau \) then

return with probability \( c \)

Let \( t = t + 1 \)

end while

Figure 2.7: Procedure to grow a track forward. Growing a track backward is done in the analogous way; e.g., \( S \) is the set of detections corresponding to \( \tau \)’s first \( s \) frames. See Section 2.3.1 for details.

and the proposed association is given by \( \omega' = (\omega\setminus\{\tau_1, \tau_2\}) \cup \tau' \).

**Split move** We choose a track \( \tau_r \) u.a.r. from \( \omega \) and pick two detections \( \tau_{r_1} \) and \( \tau_{r_2} \) u.a.r. from it, \( j_1 < j_2 \). Then, all detections before \( j_1 \) go to a new track \( \tau' \), and all detections after \( j_2 \) go to new track \( \tau'' \). Detections between \( j_1 \) and \( j_2 \) go to either track with probability \( \frac{1}{2} \). The resulting association is \( \omega' = (\omega\setminus\{\tau_r\}) \cup \{\tau', \tau''\} \). The split move is the reverse of the merge move.

**Switch move** We select two tracks \( \tau_{r_1} \) and \( \tau_{r_2} \) u.a.r. from \( \omega \), and choose one detection from each track \( \tau_{r_1j_1} \) and \( \tau_{r_2j_2} \), such that their locations on the image are within a distance \( \sigma \) times their temporal offset; that is, if \( \tau_{r_1j_1} \in B_{t_1} \) and \( \tau_{r_2j_2} \in B_{t_2} \), then \( \|\tau_{r_1j_1} - \tau_{r_2j_2}\| \leq \sigma |t_2 - t_1| \) Then, the detections after \( j_1 \) in track \( \tau_{r_1} \) and those before \( j_2 \) in track \( \tau_{r_2} \) are swapped, so that \( \tau'_{r_1} = \{\tau_{r_11}, \ldots, \tau_{r_1j_1}, \tau_{r_2j_2+1}, \ldots, \tau_{r_2|\tau_r|}\} \) and \( \tau'_{r_2} = \{\tau_{r_21}, \ldots, \tau_{r_2j_2}, \tau_{r_1j_1+1}, \ldots, \tau_{r_1|\tau_r|}\} \). The proposed association is \( \omega' = \)}
(\(\omega \setminus \{\tau_{r_1}, \tau_{r_2}\}\)) \cup \{\tau'_{r_1}, \tau'_{r_2}\}\}. The switch move is the reverse of itself.

2.3.2 Estimating the scene

Given samples \((\omega^{(i)}, \gamma^{(i)}, C^{(i)}), i = 1, \ldots, N\), we take the MAP estimate to be simply the one which has the highest posterior, i.e.,

\[
(\omega^*, \gamma^*, C^*) = \arg \max_{i=1, \ldots, N} p(\omega^{(i)}, \gamma^{(i)}, C^{(i)} | B, I)
\] (2.28)

Given these values, we can compute the MAP scene given \((\omega^*, \gamma^*, C^*)\), which, intuitively, is the scene which fits the data best, given the best data association and imaging system. That is, we want to compute a value for \(z\), which maximizes the scene posterior

\[
p(z, | \omega^*, \gamma^*, C^*, B, I) \propto p(B, I | \omega^*, z, C^*)p(z) \, dz,
\] (2.29)

or

\[
z^* = \arg \max_z p(z | \omega^*, \gamma^*, C^*, B, I).
\] (2.30)

2.4 Evaluation

So far, we have presented our approach in the general case. Later on, when we examine applications to different problems, we will discuss experiments and evaluation measures which are suitable for each task. However, since we always model target position, we can use tracking metrics in any application of our approach.

In the chapters below, we use two widely-used evaluation metrics for tracking, the CLEAR multiple object tracking (Stiefelhagen et al., 2007) metrics, and those proposed by Li et al. (2000b). The CLEAR metrics consist of two measures, multiple object tracking accuracy (MOTA) and precision (MOTP). MOTA is a measure of false positives, missed targets, and track switches, and ranges from \(-\infty\) to 1, with 1 being perfect. MOTP measures the average distance between true and inferred trajectories, and ranges from 0 to the threshold at which tracks are said to correspond, the value of which depends on the problem. These measures are designed to
measure performance at estimating the number of objects, and at keeping consistent trajectories.

Figure 2.8: A visual summary of the CLEAR metrics. Here, the blue trajectories are those estimated by the sampler, and the red trajectories are ground truth. The two trajectory portions labeled “missed” have no matching hypotheses. Conversely, the blue hypothesized trajectory on the bottom of the picture has no matching ground truth path, and is, therefore, a false positive. Finally, the blue trajectory in the middle of the picture has two different ground truth trajectories corresponding to it, which is considered a switch.

To compute them, we define the following quantities. For each frame $t$, let $m_t$ be the number of missed ground truth trajectories, $f_t$ the number of false estimated trajectories, $s_t$ the number of incorrectly matched ground truth and estimated trajectories (switches), $g_t$ the number of ground truth trajectories, $c_t$ the number correctly matched trajectories, and $d_{it}$ the distance between ground truth trajectory $\hat{x}_i$ and its corresponding estimated trajectory. Figure 2.8 depicts the computation of these values pictorially. The values for MOTA and MOTP are given by

$$\text{MOTA} = 1 - \frac{\sum_t (m_t + f_t + s_t)}{\sum_t g_t} \quad (2.31)$$
and
\[ \text{MOTP} = \frac{\sum_{i,t} d_{it}}{\sum_{i} c_t}. \] (2.32)

Note that MOTA can be negative since there is no limit on the number of false positives \( f_t \) and switches \( s_t \).

2.5 Discussion

We have shown an approach for modeling temporal scenes in world coordinates, as well as an inference paradigm for understanding them. Our approach is general, and only requires scene-specific information to be provided in order to be applied. Specifically, we need to define the world coordinate system and units, as well as the state of a target (other than position). In addition, we need to specify a camera model, as well as the accompanying transformation from world to image coordinates. Finally, we must declare the types of detections in use, as well as any other image data we wish to model. In the following chapters we discuss three specific applications of this model: tracking pollen tubes \textit{in vitro}, tracking people in 3D, and understanding head pose and gaze direction in 3D. As we shall demonstrate, this approach produces quality results in all three cases.

Let us now briefly discuss some of the main limitations of our approach and possible ways to address them. Perhaps the biggest drawback of our model is the dependence on detections for target hypotheses. That is, we model targets as appearing in the scene when they are first detected, and disappearing upon their final detection. This means that we are only robust to failures of the detector within a target’s existence. At best, this means missing targets for a few frames when they appear or disappear. If a target produces no detections, however, we will never hypothesize it (nor infer its state), no matter any other source of evidence. The obvious way to address this is by modeling the appearance and disappearance of targets separately from their detections; indeed, currently only the association-as-partition model exhibits this coupling. Naturally, the data association inference would also need extending, e.g., we need the ability to propose tracks that are not
tied to the detections. An additional concern is the fact that, in the generative model depicted in Figure 2.3b, the number of detections produced by target $r$ at time $t$ $a_{rt}$ is not generated by the state of that target $z_{rt}$. In many cases, the number of detections is directly dependent on the target’s properties, such as appearance or occlusion. Perhaps a more natural way to model this would be to have the node for $A$ placed after the node for $z$. This would create other issues, however, because of the added dependencies between variables.

The integral in Eq. 2.21 is also a potential issue, due to its high dimensionality (which grows with the length of targets’ trajectories), and the fact that the image transformation could result in non-linear components. We shall see how to handle this under these conditions in later chapters, as well as the case where the image transformation is linear.
3.1 Introduction

In this chapter we explore the problem of tracking multiple smooth trajectories\textsuperscript{1}. We are particularly interested in tracking biological tip growth as observed in tubes growing from pollen grains (see Figure 3.1). Other relevant examples which have drawn the attention of the machine vision community include seedling root growth (Miller et al., 2007), hypocotyl growth (Wang et al., 2008), and neuron growth (Al-Kofahi et al., 2006). In all these examples, there is a pressing need to quantify the trajectories automatically from image data. Typically all that can be assumed about the trajectories are that they are relatively smooth. This makes tracking very challenging when there are many trajectories that cross many times in the captured images, further compounded by the presence of noise (false points) and missing data.

We use the model described in Chapter 2, where the targets are pollen tubes growing in a petri dish (i.e., the world is 3D – the depth component is sufficiently large for tubes to cross often) and their state consists simply of their position in $\mathbb{R}^3$. The imaging system is a confocal microscope, which produces stacks of images which are taken at roughly the same time but at different focal planes. This gives us a 3D image space, where detections are points in $\mathbb{R}^3$. See Section 3.2 for more details.

In what follows, we will discuss two different questions. First, we explore the idea of using Gaussian process (GP) to model motion which is smooth, but otherwise arbitrary. We do this by applying our scene understanding model and inference approach to tracking smooth trajectories under the GP motion model and comparing it to using the standard LDS motion model. We test this model on several kinds

\textsuperscript{1}Most of the content of this chapter is the subject of our 2011 CVPR paper (Brau et al., 2011).
of smooth motion, notably tracking linear structures arising from biological tip
growth, and focusing on pollen tubes growing on a petri dish. Second, we address
the question of the effectiveness of our association sampling moves (described in
Section 2.3). To do so, we compare its performance to that of two other inference
methods: (1) the original MCMCDA, upon which our sampling moves are based,
and (2) a Gibbs sampler we developed, which takes advantage of the fact that the
integral (Eq. 2.21) has closed form, due to the linearity of the imaging system.

Before proceeding, we provide some specific motivation for tracking pollen tubes
by briefly explaining the fascinating and important science behind them. In order
for seed plants to reproduce, the male sperm cells contained in pollen grains must be
transferred to the female egg cell. Upon arrival at the surface of a flower pistil, pollen
grains absorb water from the stigma and germinate to produce pollen tubes, which
transport their cellular contents, along with two sperm cells, to the ovule of the plant
(Palanivelu and Preuss, 2006). Pollen tubes grow exclusively at their tip (Steer and
Steer, 1989), traveling through the pistil (female organ within the plant) until they
reach the ovary and enter the ovule through an opening, called the micropyle (Lord
and Russell, 2002). A pollen tube’s journey terminates at one of two synergid cells
inside the ovule, where it bursts to release sperm cells. Pollen tubes are able to
reach ovules due to the guidance provided by diffusible signals from the ovules.
Characterizing and quantifying the interaction between pollen tubes and ovules is
crucial to achieve a better understanding of the seed formation process, and could
have a very broad impact. For instance, it could lead to a better understanding of
how plants regulate fertilization and avoid spurious fertilization events, which could
be used to prevent pollen spread from genetically modified crops into native species
– a challenge breeders face while developing transgenic crop varieties.

One current difficulty in characterizing this interaction is that a human must
watch time-lapse images of pollen tubes growing in vitro (such as those in Figure
3.1), identify each tube as it grows and recognize its behavior; e.g., which pollen
tubes were targeting which ovules, which successfully entered an ovule, and which
were repelled. This process is slow and time-consuming. The first step in automating
Figure 3.1: Four image frames of a video showing pollen tubes growing \textit{in vitro}, each 10 frames apart. The green tubes are pollen tubes, each of which move toward the ovule, placed ahead of them (below in the image). These images were obtained by merging the stack of confocal images produced by the microscope.

This operation is to robustly find the paths of the pollen tubes in the images. Taking the image difference of consecutive time-lapse images provides the noisy tips of the pollen tubes at all time points, as well as plenty of background noise. Since pollen tubes exhibit smooth growth, we can apply our tracking algorithm, thereby taking a step towards modeling pollen tube growth behavior and interactions with ovules that affect tube trajectories through complex and poorly understood signaling.
3.1.1 Related work

Multiple-target tracking is a well-studied problem with many approaches. Classical approaches include the multiple hypothesis tracker (MHT) (Cox and Hingorani, 1996; Reid, 1979) and the joint probabilistic data association (JPDA) filter (Bar-Shalom et al., 1980). More recently, MCMC approaches have been proposed such as Markov chain Monte Carlo data association (MCMCDA) (Oh et al., 2004) and the MCMC-based particle filter (Khan et al., 2005). In MCMCDA, the tracking problem is solved by sampling over the space of associations of points to tracks using the Metropolis Hastings algorithm (MH). The MCMC-based particle filter also uses the MH algorithm to generate samples (particles) for the posterior at each time step. Finally, a data association approach is also used in (Huang et al., 2009), where they use Fourier inference over permutations to determine the associations. Notice, however, that all these approaches rely on motion models that are application-specific. This implies considering different dynamic models for different applications. Our goal is to provide a more general model that can be used in any application as long as the motion is smooth.

The traditional model for smooth motion is the linear-Gaussian model – also known as the linear dynamical system (LDS) (Bishop, 2006; Ghahramani and Hinton, 1996). While LDS models linear dynamics with Gaussian noise very well, it is not well suited for motion which exhibits significantly more erratic smooth behavior. Furthermore, in order to attain maximum flexibility, one is required to set many parameters. On the other hand, Gaussian processes only require a single scale parameter, assuming a chosen kernel function. Another problem with LDS models is that the state at a given time is conditionally independent of previous states given the immediately preceding state. Our model does not make this assumption, making it more flexible and able to fit a wider range of motions. We note that Gaussian processes have been used previously in tracking (Hou et al., 2007; Urtasun et al., 2006; Wang et al., 2005), but differently than in this work.
3.2 Model

As mentioned previously, we directly apply the model introduced in Chapter 2 to the problem of tracking multiple smooth trajectories. Our detections are points in $\mathbb{R}^3$, which are extracted from the video using the procedure described in Section 3.4. We then assume that the set of detections $B$ comes from an association (generated from $p(\omega)$), a scene and imaging system (generated from $p(z | \omega)$ and $p(C)$, respectively), and from the likelihood $p(B | z, \omega)$. In this chapter, we shall only discuss the scene and likelihood models as they apply to this particular problem. See Section 2.2 for a thorough exposition of all the components of our model.

3.2.1 Scene and microscope

As before, our scene is made up of targets and their states. In this case, the targets are pollen tubes, and their state is just their position. That is, for each tube $r$, its state is simply $z_r = x_r$. Although they are microscopic, pollen tubes are 3D entities, and move in the 3D world. Thus, we have that each $x_{rj} \in \mathbb{R}^3$, $r = 1, \ldots, m$, and $j = 1, \ldots, l_r$. Consequently, we need three Gaussian processes, one for each dimension, such that, for $i = 1, 2, 3$, $x_{ri} \sim \mathcal{N}(0, K_r)$, where, as before, $K_r$ is the covariance matrix, which is obtained via the covariance function. As before, pollen tubes are assumed independent, so the scene prior is given by

$$p(z | \phi_x) = \prod_{r=1}^{m} p(z_r | \phi_x). \tag{3.1}$$

We set the parameters of the GP $\phi_x = (\sigma_x, l_x)$ manually.

For this problem, our imaging system is a confocal microscope, which is able to produce stacks of images at different focal planes. This means that we have an image representation in 3D, and, under a few reasonable assumptions, we can model the imaging system $C$ as a the identity transformation. In other words, we assume that we are able to observe the pollen tubes in their world coordinates, which we will measure in pixels for simplicity.
3.2.2 Likelihood

Following the general model, each pollen tube \( r \) produces \( a_{rt} \) detections at frame \( t \), each of which is simply a Gaussian perturbation of its true position \( x_{rt} \). In other words, if \( b \) is a detection for state \( x_{rt} \), then it is distributed as \( b \mid x_{rt} \sim \mathcal{N}(x_{rt}, \text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2)) \), where \( \sigma_1^2 \), \( \sigma_2^2 \), and \( \sigma_3^2 \) are the variances for each individual dimension. We can break this down by dimension by letting \( b^i \) be the \( i \)th coordinate of \( b \), such that \( b^i \mid x_{rt}^i \sim \mathcal{N}(x_{rt}^i, \sigma_i) \), \( i = 1, 2, 3 \). Then, defining \( b_r^i \) to be the vector of the \( i \)th components of all \( b \in \tau_r \), and assuming conditional independence of detections, we have that

\[
b_r^i \mid \omega, x_r^i \sim \mathcal{N}(x_r^i, \sigma_i^2 I),
\]

where \( I \) is the identity matrix. Finally, since we assume that targets and coordinate dimensions are independent, we get a normal detection likelihood for pollen tubes, which is given by

\[
p(B \mid \omega, z, C, \phi_B) = \prod_{r=1}^{m} \prod_{i=1}^{3} p(b_r^i \mid \tau_r, x_r^i, \sigma_i),
\]

where \( \phi_B = (\sigma_1, \sigma_2, \sigma_3) \). Note the dropped \( C \) on the right hand side of Eq. 3.3. This stems from the fact that \( C \) is an identity transformation, which means that we can safely ignore it. We set all values of \( \phi_B \) manually.

3.3 Inference

As before, the distribution of interest is the posterior over variables \( (\omega, \gamma, C) \). However, recall that \( C \) is simply an identity transformation, and does not need to be inferred. Also, we note that all pollen tube videos of interest are similar in length, the number of pollen tubes and their length, as well as the amount of false alarms detected. Consequently, we will be setting \( \gamma \) manually, and consider it a constant during inference. Finally, since there are no additional sources of data other than detections, we have no \( I \) variable nor its corresponding likelihood. The posterior
distribution over which we will sample is then

\[ p(\omega \mid B) = p(\omega)p(B \mid \omega) \tag{3.4} \]

\[ = p(\omega) \int_{z} p(B \mid z, \omega)p(z \mid \omega) \, dz, \tag{3.5} \]

where \( p(\omega) \) is given by Eq. 2.14, \( p(B \mid z, \omega) \) by Eq. 3.3, and \( p(z \mid \omega) \) by Eq. 3.1. The integral in Eq. 3.5 can be computed in closed form due to the linear-Gaussian forms of \( p(B \mid z, \omega) \) and \( p(z \mid \omega) \), and we simply get that

\[ p(B \mid \omega) = \prod_{r=1}^{m} \prod_{i=1}^{3} p(b^i_r \mid \tau_r) \tag{3.6} \]

where \( b^i_r \) is the vector of the \( i \)th components of all detections associated to tube \( r \), e.g., all \( b \in \tau_r \), with \( b^i_r \mid \tau_r \sim \mathcal{N}(0, K_r + \sigma^2_i I) \).

For inference, we use the algorithm outlined in Figure 2.4, skipping the steps to sample over \( \gamma \) and \( C \). The MAP estimate of the best association is taken to be the sample \( w^* \) which has the highest posterior. To obtain an estimate for the scene \( z \) we maximize the scene posterior given \( \omega^* \) (Eq. 2.29) which, in this case, is given by

\[ p(x \mid B, \omega^*) \propto p(x \mid \omega^*)p(B \mid x, \omega^*). \tag{3.7} \]

Because we have a linear-Gaussian likelihood, this posterior is Gaussian, which implies that its max is its mean, which, for dimension \( i \) of target \( r \), can be shown to be

\[ z^*_{r,i} = x^*_{r,i} \tag{3.8} \]

\[ = \arg \max_{x^i_r} p(x^i_r \mid b^i_r, \tau^*_r) \tag{3.9} \]

\[ = K_r(K_r + \sigma I)^{-1}b^i_r, \tag{3.10} \]

and the estimate of the scene \( z^* \) is made from the components \( z^*_{r,i} \) in the obvious way.

### 3.3.1 Gibbs data association

In order to more thoroughly test the effectiveness of our inference approach, we developed an alternative association sampler which uses the Gibbs algorithm, and
to which we compare the MCMC moves discussed in Section 2.3.1. The reason Gibbs is a good fit for our problem is that we are able to compute the conditional posteriors reasonably efficiently (they are Gaussian). Before discussing the details of this sampler, we introduce an alternative formulation of an association, which is more suited for our Gibbs sampler. Let $\alpha_{tj} \in 0, \ldots, m$ denote which to track detection $b_{tj}$ belongs; that is, $\alpha_{tj} = r$ if and only if $b_{tj} \in \tau_r$. Then, we can represent an association by a vector of these assignment variables $\alpha = (\alpha)_{tj}, t = 1, \ldots, T, j = 1, \ldots, N_t$. Clearly, $\alpha$ contains the same information as $\omega$, and is, thus, an equivalent formulation.

Recall that a Gibbs sampler successively draws samples from each full-conditional distribution; in our case, the full-conditionals are given by $p(\alpha_{tj} | \alpha_{-tj}, B), t = 1, \ldots, T$ and $j = 1, \ldots, N_t$, where $\alpha_{-tj}$ is the vector $\alpha$ with variable $\alpha_{tj}$ removed, and we can sample from them as follows. First, not that, if we fix $\alpha_{-tj}$, we have that

$$p(\alpha_{tj} | \alpha_{-tj}, B) \propto p(B | \alpha)p(\alpha_{tj}, \alpha_{-tj})$$

$$= p(B | \alpha)p(\alpha)$$

$$= p(B | \omega)p(\omega)$$

We can draw sample from this distribution by first assigning equal probabilities to each possible value for $\alpha_{tj}$, i.e., $p(\alpha_{tj} = r) = (m + 1)^{-1}$, and then weighing each probability by the likelihood and prior $p(B | \omega')$ and $p(\omega')$, where $\omega'$ is the association which results from the assignment in $\alpha$, replacing $\alpha_{tj}$ with $r$. Figure 3.2 shows complete the algorithm. We can speed up the posterior computation significantly by using the fact that the likelihood is a product of the individual detection likelihoods, which means we only need to divide by the old factor and multiply by the new one.

3.4 Data, tuning, and ground truth

Let us now examine some of the details of how we obtain and preprocess our data, as well as how we tune the parameters of the model and inference. As mentioned
Algorithm 3.2: Gibbs data association

Input: Initial state $\omega^{(0)}$
Let $\alpha^{(0)}$ represent $\omega^{(0)}$

for $i = 1, \ldots, N$ do
  Let $\alpha^{(i)} = \alpha^{(i-1)}$
  for all $t = 1, \ldots, T$ and $j = 0, \ldots, N_t$ do
    for $r = 0, \ldots, m$ do
      Let $\alpha^{tj}_r = r$
      Let $p_r = \frac{1}{m+1}$
      Let $p_r = p_r p(B | \alpha') p(\alpha')$
      Let $p_r = \frac{p_r}{\sum p_r}, r = 0, \ldots, m$
      Draw $r \sim \{p_0, \ldots, p_m\}$
      Let $\alpha^{tj}_r = r$
  Set $\alpha^{(i)} = \alpha$

Samples are the corresponding $\omega^{(1)}, \ldots, \omega^{(N)}$

Figure 3.2: Gibbs sampler for data association, assuming linear-Gaussian likelihood, such as in the pollen tube case. See Section 3.3.1 for details.

previously, our initial input is a set of image stacks, with each stack corresponding to a frame, and each image slice to a different focal plane. To obtain these image, a group of pollen tubes and ovules are placed in a petri dish under a confocal microscope, and, at specified, equally-spaced time intervals, it captures the image stack by quickly taking pictures at different focal planes. The focal plane interval is roughly the width of a pollen tube, so depth information is limited.

We extract detections from these images in the following way. First, for each depth, we convert them into images of the pollen-tube tips by simply subtracting images in consecutive frames. We then perform background subtraction by subtracting two standard deviation images. Subsequently, we threshold each image using a reasonable value. Finally, we consider each image stack as a volume, and perform 26-component analysis, from which we obtain a set of 3D blobs. We take the centroid of these blobs as our detections. This simple procedure produces very good results, finding most of the pollen tube tips in each frame. Figure 3.3 contains example images of this process.
Figure 3.3: Visualization of the process by which detections are extracted from pollen tube videos. Given an input frame (a), we get images of the pollen tube tips by image differencing (b), after which we perform connected component analysis on the pixels to obtain detections (c), depicted by the red circles. See Section 3.4 for details.

The pollen tube data set contains a total of 82 image sequences, which we have separated into two groups, the “easy set”, which consists of videos with ten pollen tubes or less, and the “hard set ”, which features videos with more than ten pollen tubes. The two subsets contain 40 and 42 videos, respectively. This was done so as to obtain a more fine-grain understanding of the performance of our approach.

We also perform experiments on two other small data sets of videos found on
the Internet. Specifically, we have a birds data set, which contains four videos of birds flying in the sky, and a fish data set, which has four videos of schools of fish. These videos pose different difficulties than the pollen tube videos, such as faster target motion, and a moving camera. We obtained point detections for these videos using the method described above, with slight modifications to account for the sizes of the targets.

**Parameter tuning** In this work, the parameters of the model and inference were all set manually. For $\phi_x$, we set them to reflect a high degree of smoothness. $\phi_B$ were set to a few pixels, since the detections are not very noisy. As mentioned above, we do not sample over $\gamma$, but simply set them to reasonable values obtained from observing some of the videos. The remaining association parameter $\lambda_A$ was also tuned manually from watching the videos.

**Ground truth** For the purposes of accurately evaluating our approach, we obtained the ground truth associations for all of our data sets. We did this by manually assigning the detected blobs to the correct tracks using a point-and-click tool we developed. Given this ground truth value for $\omega$, we found the MAP scene for it using Eq. 3.10.

### 3.5 Results

We try to address two main questions with this work. First, we wish to quantify the effectiveness of our Gaussian process motion model. We do this by comparing the performance of the GP model with that of the classical model for smooth motion, the linear dynamical system (LDS), on several types of data sets which feature targets which move in a smooth, but otherwise arbitrary, way. We also wish to test the effectiveness of our inference method (Section 3.3) for data association. We compare the performance of our approach with two others: the original MCMCDA sampler introduced by Oh et al. (2004), and the Gibbs data association sampler discussed in Section 3.3.1. We measure performance in several different ways, which
we discuss next.

3.5.1 Evaluation

We use the CLEAR metrics introduced in Section 2.4, measuring MOTP in pixels. We use a threshold of 5 pixels. In addition, we develop three additional metrics. The first, which we call CTP (correct track percentage) is the percentage of detections which were assigned correctly. More precisely, given the ground truth and estimated associations $\omega$ and $\omega'$, for each pair of corresponding tracks $\tau_k$ and $\tau'_k$, CTP is given by

$$\text{CTP} = \sum_{k=1}^{K} \frac{|\tau_k \cap \tau'_k|}{|\tau_k|},$$

(3.14)

where we assign each estimated track to the ground truth track with which it shares the most detections. Although this is similar to MOTA, it differs in that CTP compares against the ground truth data association, whereas MOTA computes the correspondences based on trajectory distance.

The second metric, called curve distance (CD), is an evaluation based on the similarity of two corresponding trajectories. Specifically, given two corresponding true and estimated tracks $\tau_r$ and $\tau'_r$, their curve distance is

$$C_r = \sum_{j=1}^{l_r} \frac{(x_{rj} - x'_{rj})^2}{l_r}.$$  

(3.15)

(Here, we assume that $\tau_r$ and $\tau'_r$ are the same length; when it is not the case, we simply ignore the non-matching locations.) The CD error is then calculated as $\text{CD} = \frac{1}{m} \sum_{r=1}^{m} C_r$, and is normalized to the size of the image. This metric is similar to MOTP, but differs in one key way. MOTP measures distance over all matching trajectory points, whereas CTP considers complete trajectories as a whole.

For the last evaluation, we compute the average error committed in the estimate of the number of tracks in an image sequence. In other words, if $\omega$ and $\omega'$ are a ground truth and a estimated association (respectively), the error for that image sequence is simply $K_{\text{err}} = ||w| - |w'||$. 

3.5.2 Gaussian process as smooth motion

One of the main objectives of this work is to show that a Gaussian-process-based motion model is useful for multiple-target tracking, under the smooth trajectory conditions. In order to do this, we must compare our model with others that are useful for similar problems, such as LDS. To this end, we have performed the tests discussed below. For the LDS tests, we used a very typical setup, where motion is assumed to be linear and noise is Gaussian. Specifically, the state of target $r$ at frame $t$, $x_{rt} \in \mathbb{R}^6$ consists of position and velocity, and is given by

$$ x_{rt} = B x_{rt-1} + w, $$  \hfill (3.16)

where

$$ B = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, $$  \hfill (3.17)

and $\Delta t$ is the time elapsed between $x_{rt}$ and $x_{rt-1}$. Each $x_{rt}$ produces a detection $b_{tj} \in \mathbb{R}^3$ which is given by

$$ b_{tj} = H x_{rt} + v, $$  \hfill (3.18)

where

$$ H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. $$  \hfill (3.19)

Here, $v \in \mathbb{R}^6$ and $w \in \mathbb{R}^3$ are normally-distributed random variables. Assuming $x_{r1}$ is normally distributed, the posterior distribution $p(\omega \sim B)$ will be a normal distribution.

We compare our GP motion model to LDS by running our tracker on the data sets described in Section 3.4 using both models. Specifically, we ran the tracker on
Table 3.1: A comparison between the performance obtained using the Gaussian process model and that of the LDS model on the two pollen tube data sets. In the table above, MCMCDA-GP is the MCMCDA algorithm using the GP model and MCMCDA-LDS uses the LDS model. The analogous terminology was used for GDA. We show values for CTP (correct track percentage, bigger is better), CD (curve distance, smaller is better), and $K_{err}$ (smaller is better). All values in the tables are averages over the set of image sequences in each data set. See text for details on these measures. In general, the Gaussian process model outperforms LDS using both trackers, and GDA-GP tracker gives the best results overall.

<table>
<thead>
<tr>
<th></th>
<th>Easy Set</th>
<th></th>
<th>Hard Set</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CTP</td>
<td>CD</td>
<td>$K_{err}$</td>
<td>CTP</td>
</tr>
<tr>
<td>MCMCDA-GP</td>
<td>0.87</td>
<td>0.05</td>
<td>0.31</td>
<td>0.79</td>
</tr>
<tr>
<td>MCMCDA-LDS</td>
<td>0.71</td>
<td>0.09</td>
<td>0.52</td>
<td>0.47</td>
</tr>
<tr>
<td>GDA-GP</td>
<td>0.93</td>
<td>0.01</td>
<td>0.22</td>
<td>0.88</td>
</tr>
<tr>
<td>GDA-LDS</td>
<td>0.74</td>
<td>0.07</td>
<td>0.61</td>
<td>0.59</td>
</tr>
<tr>
<td>Heuristic</td>
<td>0.51</td>
<td>0.28</td>
<td>1.83</td>
<td>0.13</td>
</tr>
</tbody>
</table>

The pollen tube videos using both MCMCDA and GDA inference to compare motion models vis-a-vis inference method, and on the birds and fish videos using only GDA to test on different types of motion. Table 3.1 summarizes the first experiment, where we can see that using the GP model consistently gives better performance in both the “easy” and “difficult” videos.

Table 3.2: Results of applying our Gibbs tracker to four videos of schools of fish and four videos of flocks of birds. GDA-GP outperforms GDA-LDS even though the motion of these targets can be well modeled in 3D by an LDS.

<table>
<thead>
<tr>
<th></th>
<th>CTP</th>
<th>CD</th>
<th>$K_{err}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDA-GP</td>
<td>0.89</td>
<td>0.10</td>
<td>1.33</td>
</tr>
<tr>
<td>GDA-LDS</td>
<td>0.77</td>
<td>0.11</td>
<td>2.16</td>
</tr>
</tbody>
</table>

In Table 3.2, we see that the same holds true for very different types of (smooth) motion. It is worth noting that fish and birds largely exhibit linear motion, which is exactly the type of motion that LDS is meant to model. In spite of this, the GP model gives better results than the LDS model. We propose that, while these types of targets do move according to an LDS, their images on the camera do not. There are two main reasons for this: (1) the targets are being projected on the image plane
and (2) the movement of the camera which took the images. Clearly, a projection from 3D to 2D completely changes the path that, say, a fish might take. Also, if the camera itself is moving as the video is shot, the relative motion of the targets is drastically altered. The motion is nevertheless still smooth, and thus the Gaussian process model works reasonably well. Figures 3.4 and 3.5 show some qualitative results of these experiments.

Figure 3.4: An example of a sequence from the difficult set. This association has 60 tracks, of which GDA-GP found 53, all almost completely correct. By contrast, GDA-LDS only found 35 most of which were not completely on target, and some of which were off by a wide margin.

3.5.3 Inference

To test the effectiveness of our inference method, we compare it to two other approaches, the classical MCMCDA and the Gibbs data association detailed in Section 3.3.1. We ran our tracker on the pollen tube data sets using the GP motion model with all three inference algorithms, and Table 3.3 has the results. Our method performs the best on all data sets, with GDA coming in a close second. Note the
difference in performance when using our approach to that of MCMCDA, which use very similar methods. The main difference between these two samplers is that MCMCDA only allows each target to be assigned a single detection per frame, whereas ours is more flexible. This matches our intuition that having multiple detections per track is beneficial, both in the model and during inference.

<table>
<thead>
<tr>
<th></th>
<th>Easy Set</th>
<th>Hard Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MOTA</td>
<td>MOTP</td>
</tr>
<tr>
<td>MCMCDA</td>
<td>0.69</td>
<td>4.4</td>
</tr>
<tr>
<td>GDA</td>
<td>0.81</td>
<td>4.6</td>
</tr>
<tr>
<td>Ours</td>
<td>0.85</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Table 3.3: Comparison of different inference approaches. We ran three inference algorithms – MCMCDA, GDA, and our variation of MCMCDA – on the pollen tube data sets, and evaluated using the CLEAR metrics for multiple-object tracking, with a threshold of 5 pixels. The best performance was achieved by our approach, and MCMCDA performed the worst.

3.6 Discussion

We presented an application of our approach for temporal scene understanding to the problem of tracking multiple smooth trajectories. We showed the effectiveness of our smooth motion model by comparing it with a more standard model, the linear dynamical system, by integrating it into two different tracking approaches, and testing it on two diverse data sets. The results suggest that this is a very
effective way to follow multiple, overlapping tracks, which are relatively smooth, but otherwise arbitrary. Further, the combination of data association and Gaussian processes seems particularly effective, partly by being able to exploit the full run of data if it is available. Even when the number of tracks was large (hard pollen tube data set), only one of them was missed on average, and the ones that were found were followed relatively accurately. We also showed the performance of our data association inference algorithm by testing it against two methods, the well-known MCMCDA algorithm by Oh (2008) and a Gibbs sampler which we developed specifically for this problem. Our approach outperforms both MCMCDA and our Gibbs sampler in the two pollen tube data sets.

Although these results are encouraging, it is important to keep in mind that this scenario has two properties which make it simpler than other typical scene understanding problems. First, the imaging system is the identity transformation, which means that the likelihood is simple to compute. The physical state of the tracked objects is also very simple, as it only includes the position. As we shall see, both of these properties will be lost in our other applications, making the problem significantly more challenging. Nevertheless, we emphasize that our algorithm performs very well in this scenario.
CHAPTER 4

TRACKING PEOPLE IN 3D

4.1 Introduction

Let us discuss another application of our approach, this time to tracking people in 3D and estimating their size from monocular video. Tracking people in videos is a widely-studied problem, and has many applications in areas such as surveillance and autonomous driving. Given a video, the objective is to locate each person and track their movement in the scene. Our goal is to do recover this information in 3D, as well as to estimate the size of each person and the camera parameters.

This problem presents several additional challenges when compared to that of tracking pollen tubes. First, the imaging system is not known beforehand, and must be estimated during inference. Second, people move in a wider variety of ways than do pollen tubes, and, in some videos, occlude each other much more severely, due to different camera and scene configurations. People also have a much more complex physical state, including size and pose, as opposed to points in pollen tubes. Finally, there is a rich variety of situations, from videos of pedestrians in an outdoor scene, to people interacting in a room or lab.

In the following sections, we will examine the components of our model as applied to this problem. We model people as upright 3D cylinders moving on the ground plane, whose motion is described by a Gaussian process. We use the perspective camera representation to project the scene onto the image plane, and model detections as noisy bounding boxes around the projected cylinders, and optical flow features as noisy displacement vectors of consecutive cylinder projections. For inference, we use the approach detailed in Section 2.3. We will also discuss our method for approximating the integral over scenes, which, due to the perspective projection
and non-Gaussian likelihood, does not have closed form.

We test our method on benchmark videos, and compare our performance with some of the current state-of-the-art trackers. Our experiments show performance which is comparable to other trackers using standard evaluation measures. It is worth noting that we estimate the camera and the size of each person, whereas other trackers do not.

4.1.1 Related work

Our data association approach extends that of Oh et al. (2004). We further follow Brau et al. (2011) who used Gaussian processes for trajectory smoothness while searching over associations by sampling. Wu et al. (2011) use a similar data association model, but propose an effective non-sampling approach for inference. All these efforts are focused on association of points alone; neither appearance or geometry are considered. With respect to representation, several other efforts share our preference for 3D Bayesian models for tracking (e.g., Sidenbladh et al. (2000); Choo and Fleet (2001); Sminchisescu and Triggs (2003)).

Wojek et al. (2010) improve a tracking-by-detection approach using 3D geometric reasoning methods introduced by Hoiem et al. (2006). They assume the camera is known, but model small deviations in pitch. They also link detections with an association model, but do not consider smoothness.

Andriyenko and Schindler (2010) pose data association as an integer linear program (following work by Berclaz et al. (2009)), by placing collision avoidance and continuity constraints on their tracks. In subsequent work, Andriyenko and Schindler (2011) formulate an energy approach for multi-target tracking in 3D that includes terms for image evidence, physics based priors, and a simplicity term that pushes towards fewer trajectories. Although, much of the insight in that work has analogs in our probabilistic formulation, they do not explicitly deal with data association.

Later, Andriyenko et al. (2012) attempt to solve both data association and trajectory estimation problems simultaneously, using similar modeling ideas as in their
previous work. In contrast to our work, they attempt to optimize both association and trajectory energy functions, which results in a space of varying dimensionality, making optimization more difficult.

Our approach to inferring the camera conditioned on object hypotheses has a lot in common with recent work on scene understanding from single view (Del Pero et al., 2011) and multiple views (Bao and Savarese, 2011). An approach related to ours, but applied to pose, is to find 2D tracklets and then infer 3D information using a 3D model (Andriluka et al., 2010).

Khan et al. (2006) use a data association model which allows for multiple detections per target. However, the different detections correspond to different parts of the target, whereas we model multiple detections as arising from using multiple detectors. Finally, while we advocate a 3D approach to understanding occlusion, there is also recent work on modeling occlusion in 2D (Kwak et al., 2011).

4.2 Model

We use the Bayesian generative formulation discussed in Chapter 2. We have two sources of data, person detectors and optical flow features (see Section 4.4 for details on how we obtain it), which we assume to be the result of the process implied by our generative model. First, an association object $\omega$ is sampled from the prior $p(\omega)$. Then, the 3D scene $z$ and camera $C$ are generated from $p(z)$ and $p(C)$, respectively. Each cylinder in the scene is projected onto the image plane via the camera, resulting in a set of bounding boxes and displacement vectors, which are perturbed to generate our data detections $B$ and flow features $I$, according to the likelihood function $f(B, I | \omega, z, C)$. We examine these in detail in the following sections, except the model for data association, which is discussed in Section 2.2.1.

4.2.1 Scene and camera

We model the scene as a set of people in 3D. The scene $z$ consists of each individual person's state $z_r$, $r = 1, \ldots, m$. In turn, each person's state can be decomposed into
their trajectory $\mathbf{x}_r = (x_{r1}, \ldots, x_{rl})$ and their physical state, which is composed of their height, width, and girth, i.e., $\mathbf{w}_r = (h_r, w_r, g_r)$. We can think of each person as being represented by a $h_r \times w_r \times g_r$ cylinder that moves on the ground plane. We assume the trajectory of each person is a realization of a Gaussian process with a kernel which promotes smooth motion. Under this model, we have that

$$p(\mathbf{x}_r) = \prod_{i=1}^{D_x} p(\mathbf{x}_r^i),$$

where $\mathbf{x}_i$ is the vector of the $i$th components of $\mathbf{x}_r$, and $D_x = 2$, since the ground plane is 2D. Each individual prior $p(\mathbf{x}_r^i)$ is the Gaussian pdf which results from the Gaussian process. See Section 2.2.2 for more details. Person size is a priori normally distributed, i.e.,

$$h_r \sim \mathcal{N}(\mu_h, \sigma_h),$$
$$w_r \sim \mathcal{N}(\mu_w, \sigma_w),$$
$$g_r \sim \mathcal{N}(\mu_g, \sigma_g).$$

As before, the prior over a person’s state is $p(\mathbf{z}_r) = p(\mathbf{x}_r)p(\mathbf{w}_r)$, and the full scene prior is given by

$$p(\mathbf{z} | \phi_{\mathbf{x}}, \phi_{\mathbf{w}}) = \prod_{r=1}^{m} p(\mathbf{z}_r | \phi_{\mathbf{x}}, \phi_{\mathbf{w}}),$$

The parameters $\phi_{\mathbf{x}}$ are set from data (see Section 4.4), and $\phi_{\mathbf{w}} = (\mu_h, \sigma_h, \mu_w, \sigma_w, \mu_g, \sigma_g)$ are set manually, chosen to follow actual human size, whose distribution is detailed by McDowell et al. (2005).

We assume a standard perspective camera (Hartley and Zisserman, 2000) with some simplifying assumptions (Del Pero et al., 2011). We set the origin of the world to be on the ground plane, for which we use the $xz$-plane. We assume the camera center to be at $(0, \eta, 0)$ ($\eta$ is the camera height), a pitch angle of $\psi$, and a focal length of $f$ (see Figure 4.1). Further, we assume the camera has unit aspect ratio, and that the roll, yaw, axis skew, and principal point offset are all zero. We let $\eta$,
ψ, and f have vague normal priors, i.e., we have

\[ \eta \sim \mathcal{N}(\mu_{\eta}, \sigma_{\eta}), \tag{4.6} \]
\[ \psi \sim \mathcal{N}(\mu_{\psi}, \sigma_{\psi}), \tag{4.7} \]
\[ f \sim \mathcal{N}(\mu_f, \sigma_f). \tag{4.8} \]

Assuming independence between parameters, the camera prior is

\[ p(C \mid \phi_C) = p(\eta \mid \mu_\eta, \sigma_\eta)p(\psi \mid \mu_\psi, \sigma_\psi)p(f \mid \mu_f, \sigma_f), \tag{4.9} \]

where \( C = (\eta, \psi, f) \) and \( \phi_C = (\mu_\eta, \sigma_\eta, \mu_\psi, \sigma_\psi, \mu_f, \sigma_f) \) are the prior parameters which are set manually and are vague.

Figure 4.1: Model box computation. The cylinder from target \( z_r \) in frame \( j \) gets projected via camera onto the image plane, and model box \( h_{rj} \) is computed around it. The camera has height \( \eta \) and pitch \( \psi \).

**Scene projection**

Given the scene and the camera, we must project it onto the image plane in order to ultimately generate the detections \( B \) and image data \( I \). We do this by projecting each person at each frame in the following way. For a given person \( r \) and their \( j \)th position, we sample points on the surface of cylinder \( z_{rj} \), and project them using the camera \( C \). We find a tight bounding box \( h_{rj} \) around these projected image points, which we call a *model box*. In this way, every person has a corresponding model box.
at each frame in which they are in the scene. Figure 4.1 illustrates this process. For each model box $h_{rj}$, we also compute the region $\hat{h}_{rj}$ that is not occluded from the camera as follows. First, we discretize $h_{rj}$ into a grid of small cells. We then shoot a ray from the center of each grid cell to the center of the camera, and declare it visible if the ray does not intersect any other box. Then, $\hat{h}_{rj}$ is simply the union of these visible cells. Figure 4.2 illustrates this idea pictorially.

![Diagram](image)

Figure 4.2: Occlusion reasoning. Two model boxes, $h_{rj}$ and $h_{r'j'}$, are computed for persons $z_{rj}$ and $z_{r'j'}$ (see Figure 4.1). The visible region $\hat{h}_{r'j'}$ of box $h_{r'j'}$ is the area which does not intersect model box $h_{rj}$. In this case, since $h_{rj}$ is not occluded, $h_{rj} = \hat{h}_{rj}$.

We also compute the 2D direction of each person at each frame. This is used to compute the image likelihood, which is based on optical flow features. Given two consecutive model boxes for the same person $h_{rj}$ and $h_{rj+1}$, the model direction $u_{rj}$ of $h_{rj}$ is the vector from the center of $h_{rj}$ to the center of $h_{rj+1}$. See Figure 4.3b for an example of this.

4.2.2 Likelihood

We use two sources of evidence: person detectors and optical flow. First, we run various person detectors on the video frames to get bounding boxes $B_t = \{b_{t1}, \ldots, b_{tN_t}\}$, $t = 1, \ldots, T$, where $N_t$ is the number of detections in frame $t$. We parametrize each box $b_{tj}$ by $(b_{tj}^x, b_{tj}^{top}, b_{tj}^{bot})$, representing the $x$-coordinate of the center, and the $y$-
coordinates of the top, and bottom, respectively. We also run a dense optical flow estimator on the video, which outputs a set of velocity vectors \( I_t = \{v_{t1}, \ldots, v_{tN_I}\} \) for each frame \( t = 1, \ldots, T - 1 \), where \( N_I \) is the number of pixels in the frame. Finally, we use \( B = \bigcup_{t=1}^{T} B_t \) and \( I = \{I_1, \ldots, I_{T-1}\} \).

Detection likelihood

The set of detection boxes \( B \) can be partitioned into two disjoint sets, the set of false alarms \( \tau_0 \) and the set of “true” detections \( B \setminus \tau_0 = \bigcup_{r=1}^{m} \tau_r \). For the latter, the likelihood is defined as follows. Let \( b \in B \tau_r \) be a detection box assigned to person \( r \), and let \( h_{rj} \) be its corresponding model box. We assume that \( b \) has i.i.d. Laplace-distributed errors in the \( x \), top, and bottom parameters. That is, for parameter \( b^x \), we have that

\[
    b^x - h_{rj}^x \sim \text{Laplace}(\mu^x, \sigma^x),
\]

which implies that

\[
    b^x | h^x \sim \text{Laplace}(h^x + \mu^x, \sigma^x).
\]

Figure 4.3a illustrates this distribution. Analogously, we have that both

\[
    b^{\text{top}} | h^{\text{top}} \sim \text{Laplace}(h^{\text{top}} + \mu^{\text{top}}, \sigma^{\text{top}}),
\]

\[
    b^{\text{bot}} | h^{\text{bot}} \sim \text{Laplace}(h^{\text{bot}} + \mu^{\text{bot}}, \sigma^{\text{bot}}).
\]

As stated above, we assume that \( b^x, b^{\text{top}}, \) and \( b^{\text{bot}} \) are independent, giving us, for all \( b \in B \setminus \tau_0 \), a single detection likelihood of

\[
    p(b | h, \phi_B) = p(b^x | h^x, \mu^x, \sigma^x)p(b^{\text{top}} | h^{\text{top}}, \mu^{\text{top}}, \sigma^{\text{top}})p(b^{\text{bot}} | h^{\text{bot}}, \mu^{\text{bot}}, \sigma^{\text{bot}}),
\]

where \( h \) is the model box for \( b \), and with \( \phi_B = (\mu^x, \sigma^x, \mu^{\text{top}}, \sigma^{\text{top}}, \mu^{\text{bot}}, \sigma^{\text{bot}}) \). If \( b \) is assigned to a model box which is not fully visible, we consider \( b \) to be partially a false alarm. That is, the likelihood for \( b \) is actually

\[
    p(b | h, \phi_B, w_I, h_I) = \hat{h} | p(b | h, \phi_B) + (1 - \hat{h}) | p(b | w_I, h_I),
\]
where \( p(b) \) is the false alarm detection likelihood, described below, and \(|\widehat{h}|\) is the area of \( \widehat{h} \), i.e., the fraction of \( h \) which is visible.

We model false alarm detections as being uniformly distributed across the image. That is, for all \( b \in \tau_0 \), we have that \( p(bx) = \frac{1}{w_I} \), \( p(b^{\text{top}}) = \frac{1}{h_I} \), and \( p(b^{\text{bot}}) = \frac{1}{h_I} \), where \( w_I \) and \( h_I \) are the width and height of the image. As before, we treat the three parameters as independent, which gives us a single detection false alarm likelihood of

\[
p(b \mid w_I, h_I) = p(bx \mid w_I)p(b^{\text{top}} \mid h_I)p(b^{\text{bot}} \mid h_I), \tag{4.16}
\]

for all \( b \in \tau_0 \). Combining all these factors, and considering conditional independence, we get a detection likelihood given by

\[
p(B \mid z, \omega, C, \phi_B, w_I, h_I) = \prod_{b \in \tau_0} p(b \mid w_I, h_I) \prod_{b \in B \setminus \tau_0} p(b \mid h(b), \phi_B, w_I, h_I), \tag{4.17}
\]

where \( h(b) \) is the model box of the cylinder for the target and frame corresponding to box \( b \). Distribution parameters \( \phi_B \) are set from data (see Section 4.4 for details).

**Image likelihood**

We aggregate optical flow vector into averages as follows. Let \( I_B \) be the set of boxes of all sizes and locations that fit within the image, and \( \bar{v}_t(b) \) be the average of the optical flow vectors from frame \( t \) contained in box \( b \). We redefine \( I_t = \{ \bar{v}_t(b) \mid b \in I_B \} \), and let \( I = \{ I_1, \ldots, I_{T-1} \} \) as before. Given a 3D scene \( z \), we can partition each \( I_t \) into two sets: the set of foreground vectors \( I_t^* = \{ v \in I_t \mid v \text{ is the model vector for a model box of } z \text{ at frame } t \} \), and the set of background vectors \( I_t \setminus I_t^* \). For every \( v = (v^x, v^y) \in I_t \) and corresponding model box \( h_{rj} \), we model the error between each of their coordinates as having a Laplace distribution, so that \( v^x \mid u_{rj} \sim \mathcal{L} \left( u_{rj}^x, \sigma_{x}^r \right) \) and \( v^y \mid u_{rj} \sim \mathcal{L} \left( u_{rj}^y, \sigma_{y}^r \right) \), where \( u_{rj} \) is the model direction of \( h_{rj} \) (see Figure 4.3b). Further, in order to deal with occlusion properly, we redefine \( v \in I_t^* \) as the average of the optical flow vectors in the visible region of each box. Considering \( v^x \) and \( v^y \) to be conditionally independent,
Figure 4.3: Pictorial representation of the likelihood function. On the left (a), we compare model box $h_{r_j}$ (in blue) with corresponding detection box $b_{r_j}$ (in red), each represented by their top, their bottom, and their center ($h_{r_j}^\text{top}$, $h_{r_j}^\text{bot}$, and $h_{r_j}^x$, respectively). This example shows the Laplace distribution around the model box $x$-center. For the image likelihood (b), model direction $u_{r_j}$ (blue arrow) is compared against $v$ (red arrow), the average of all the flow vectors inside $\hat{h}_{r_j}$, which only includes the visible part of $h_{r_j}$ (blue box), since it is partially occluded by the red model box, presumably belonging to a different person.

we get a single-vector likelihood of

$$p(v \mid u_{r_j}, \phi_I) = p(v^x \mid u_{r_j}^x, \sigma_I^x)p(v^y \mid u_{r_j}^y, \sigma_I^y),$$

(4.18)

for all $v \in I^*$, where, again, $u_{r_j}$ is the model direction corresponding to data vector $v$, and $\phi_I = (\sigma_I^x, \sigma_I^y, \tilde{\sigma}_I^x, \tilde{\sigma}_I^y)$.

For background vectors, $v \in I \setminus I^*$, we put vague Laplace likelihoods on their coordinates; i.e., $v^x \sim \text{Laplace}(0, \tilde{\sigma}_I^x)$ and $v^y \sim \text{Laplace}(0, \tilde{\sigma}_I^y)$. As above, we assume independence, and we have that $p(v) = p(v^x \mid \tilde{\sigma}_I^x)p(v^y \mid \tilde{\sigma}_I^y)$. We put all this together to get the full image likelihood, which is given by

$$p(I \mid z, \omega, C, \phi_I) = \prod_{t=1}^{T-1} \left[ \prod_{v \in I^*_t} p(v \mid u(v), \phi_I) \prod_{v \in I_t \setminus I^*_t} p(v \mid \phi_I) \right],$$

(4.19)

where $u(v)$ is the model direction corresponding to $v$. We can simplify this expression by taking advantage of the sparsity of the trajectory boxes and dividing by the
constant $\prod_{v \in I} p(v \mid \phi_I)$ to get

$$p(I \mid z, \omega, C, \phi_I) \propto \prod_{t=1}^{T-1} \prod_{v \in I_t^*} \frac{p(v \mid u(v), C, \phi_I)}{p(v \mid \phi_I)}.$$  \hspace{1cm} (4.20)

Similarly to the detection likelihood, we calibrate the distribution parameters $\phi_I$ from data. This process is detailed in Section 4.4.

### 4.3 Inference

We wish to find the MAP estimate of $\omega$ as a good solution to the data association problem (see Section 2.3). In addition, we need to infer the camera parameters $C$, and the association prior parameters $\gamma = (\kappa, \theta, \lambda_N)$, which we consider functions of the video. Hence, we seek a value $(\omega, C, \gamma)$ that maximizes the posterior distribution

$$p(\omega, \gamma, C \mid B, I) = p(\omega \mid \gamma)p(\gamma)p(C)p(B, I \mid \omega, \gamma, C)$$  \hspace{1cm} (4.21)

$$= p(\omega \mid \gamma)p(\gamma)p(C) \int z p(B, I \mid \omega, z, C)p(z) \, dz,$$  \hspace{1cm} (4.22)

where the factors in the expression are given by equations 2.14, 2.15, 4.9, 4.17, 4.20, and 4.5. We use the inference method outlined in Section 2.3 (see also Figure 2.4). That is, we use Gibbs sampling and successively draw samples from the three full-conditional distributions $p(\gamma \mid \omega)$, $p(\omega \mid \gamma, C, B, I)$, and $p(C \mid \omega, B, I)$. In what follows, we discuss each of these steps, as well as how to compute the integral over scenes, since it does not have closed form. We use the Hamiltonian Monte Carlo algorithm for both of these tasks, which is discussed in some detail in Section B.4.

#### Sampling associations and their parameters

We use Gibbs to draw samples of the association parameters $\gamma$ from the conditional posterior $p(\gamma \mid \omega)$, using equations 2.25, 2.26, and 2.27. Similarly, we apply the approach described in Section 2.3.1 to sample associations from $p(\omega \mid \gamma, C, B, I)$. Since the association moves are identical, we shall not discuss them here. However, We note that the detections are now boxes, and their image plane location is given
by their bottom center point. Figure 4.4 shows the sampling moves for association as applied to detection boxes, and Figure 4.5 illustrates the track-growing procedure when using detection boxes.

Figure 4.4: Illustration of the seven MCMC proposal moves we use. Each figure contains an example a move, with the arrow indicating the direction of the move (i.e., $\omega \rightarrow \omega'$). Detections are boxes on the image plane, and the numbers inside indicate in which frame they exist. Black boxes represent false alarms, and different colored boxes represent different tracks (red and blue, in this case).
Figure 4.5: Illustration of the procedure to grow a track forward. The red boxes indicate $S$, the set of detections for the last $s$ frames of the growing track $\tau$, and the black boxes are the set of available detections at frame $t$, $\tau_0 \cap B_t$, which are all candidates to be appended to $\tau$. The red line is fit to the detections, and each detection in $\tau_0 \cap B_t$ is added to $\tau$ with probability given by the Gaussian surface placed around the predicted point on the line, represented by the concentric circles on the diagram. See Section 2.3.1 and Figure 2.7 for more details.

**Sampling cameras**

To draw samples from the camera conditional posterior, given by

$$p(C \mid \omega, \gamma, B, I) \propto p(C)p(B, I, \mid \omega, C).$$

we use the Hamiltonian Monte Carlo algorithm (HMC), as this has proved effective in the task of camera estimation under a similar parametrization (see Del Pero et al. (2011)). HMC is a MCMC algorithm which uses the gradient of the target distribution to explore the solution space more efficiently. In particular, it successfully avoids the random walk behavior exhibited by Metropolis-Hastings (see Section B.4 for further details). We apply the algorithm outlined in Figure B.3, where the target distribution is Eq. 4.23. Specifically, at each iteration of the algorithm, we sample a momentum variable $p \sim \mathcal{N}(0, I)$, and evolve dynamics on both $p$ and the current
state $C^{(i-1)}$ to get proposed state $p' C'$, which is accepted with probability

$$
\min \left( 1, \frac{p(C' | \omega, B, I) f(p')}{p(C | \omega, B, I) f(p)} \right),
$$

(4.24)

where $f$ is the multivariate normal pdf. We initialize the sampler at the mean of the prior $C^{(0)} = (\mu_\eta, \mu_\psi, \mu_f)$.

In order to use HMC, we must be able to compute the gradient of the log-posterior. Due to the fact that the posterior contains a complex integral, we cannot differentiate it analytically. Instead, we approximate it using finite differences. For example, the component of the gradient vector corresponding to the camera height $\eta$ is approximated in the following way:

$$
\frac{\partial}{\partial \eta} \log p(C | \omega, B, I) \approx \frac{\log p(C + \Delta \eta | \omega, B, I) - \log p(C - \Delta \eta | \omega, B, I)}{2\Delta \eta},
$$

(4.25)

where $C + \Delta \eta$ is a camera configuration equal to $C$, but with the $\eta$ parameter changed by $\Delta \eta$, which is a small value. The other two gradient components, $\frac{\partial}{\partial \psi} \log p(C | \omega, B, I)$ and $\frac{\partial}{\partial f} \log p(C | \omega, B, I)$, are computed in the analogous way.

**Scene marginalization**

As mentioned before, the evaluation of the posterior (Eq. 4.23) requires the evaluation of the likelihood integral over scenes

$$
p(B, I | \omega, \gamma, C) = \int_z p(B, I | \omega, z, C)p(z) dz,
$$

(4.26)

The same holds for the association posterior (Eq. 2.20). Due to the camera projection, this likelihood cannot be performed analytically, nor can it be computed numerically, due to the high dimensionality of $z$. Instead, we estimate the value of the likelihood in the following way. First, we note that

$$
p(B, I | \omega, C) = \frac{p(B, I | z, \omega, C)p(z)}{p(z | B, I, \omega, C)},
$$

(4.27)

holds for any scene $z$. (This equality is known as Candidate’s formula.) In particular, it holds when $z$ is set to the MAP

$$
z^* = \arg \max z p(z | B, I, \omega, C),
$$

(4.28)
that is,
\[
p(B, I \mid \omega, C) = \frac{p(B, I \mid z^*, \omega, C)p(z^*)}{p(z^* \mid B, I, \omega, C)}.
\] (4.29)

Unfortunately, we can only compute the denominator of this fraction (i.e., the scene posterior) up to a constant factor. Consequently, we approximate with a Gaussian pdf using Laplace’s method, which is given by
\[
p(z^* \mid B, I, \omega, C) \approx (2\pi)^{-\frac{D_z}{2}}|H|^{-\frac{1}{2}} \exp \left(-\frac{1}{2}(z^* - z^\star)^\top H(z^* - z^\star)\right)
\] (4.30)
\[
= (2\pi)^{-\frac{D_z}{2}}|H|^{-\frac{1}{2}},
\] (4.31)

where \(D_z\) is the dimensionality of \(z\), \(H\) is the Hessian of \(-\log p(B, I \mid z, \omega, C)p(z \mid \omega)\) evaluated at \(z^\star\). Using this, we get an approximation for the likelihood in Eq. 4.26 of
\[
p(B, I \mid \omega, C) \approx (2\pi)^{-\frac{D_z}{2}}|H|^{-\frac{1}{2}} p(B, I \mid z^*, \omega, C)p(z^*).
\] (4.32)

This is called the Laplace-Metropolis approximation of the marginal likelihood. An exact computation of \(H\) is intractable, since it requires \(O(D_z^2)\) evaluations of the posterior (eq. 4.33). Consequently, we only compute a subset of its elements, taking into account conditional independence between variables. See Section A.2 for the details.

**Finding \(z^\star\)** We again use the HMC algorithm to find \(z^\star\). In this case, our target distribution is the scene posterior
\[
p(z \mid B, I, \omega, C) \propto p(z)p(B, I \mid z, \omega, C).
\] (4.33)

We again simply apply algorithm B.3. The initial state of the sampler \(z^{(0)}\) is obtained in the following way. For the trajectories \(x^{(0)}\), we back-project the bottoms of the detection boxes \(\tau_r\) associated to each person \(r\) to the 3D ground plane using the current camera estimate, and smooth the back-projected points. The initial value for \(h_r^{(0)}\) is obtained by back-projecting the tops of \(\tau_r\), and, along with the back-projected bottoms, computing 3D heights for each frame. We then set \(h_r^{(0)}\) to the
median of these per-frame heights. The width and girth are simply set to the mean of their respective priors, i.e., $w_r^{(0)} = \mu_w$ and $g_r^{(0)} = \mu_g$.

Using HMC also requires us to compute the gradient of the log-posterior. Since it cannot be computed exactly, we must approximate it, which we do using central finite differences. Specifically, the $i$th component of the gradient vector is given by

$$\frac{\partial}{\partial z_i} \log p(z \mid B, I, \omega, C) \approx \frac{\log p(z + \varepsilon_i e_i \mid B, I, \omega, C) - \log p(z - \varepsilon_i e_i \mid B, I, \omega, C)}{2\varepsilon_i},$$

(4.34)

where $z_i$ is the $i$th component of $z$, $\varepsilon_i$ is the step size for the $i$th component, and $e_i$ is the $i$th standard basis unit vector, e.g., $e_1 = (1, 0, \ldots, 0)$. Again, this computation could potentially be prohibitively expensive, due to the high number of computations of the posterior. However, our model exhibits a lot of independence, which we exploit in order to speed up computation time significantly. Section A.2 contains the details.

4.4 Data, tuning, and ground truth

In this section, we discuss how we obtain and preprocess our data, as well as how we calibrate and tune parameters of the model and inference. As mentioned above, the initial input to our algorithm is a video in any standard format. Using standard software, we extract the frames from the video as images. From these images, we extract person detections and optical flow features. For person detections, we use the readily available MATLAB implementation of the object detector developed by Felzenszwalb et al. (2009), pre-trained for humans. See Figure 4.6 for a typical output of the detector. We found that the detector missed well-defined smaller figures, mitigated by using double-sized images. For the image data, we precomputed the dense optical flow of each frame using an existing software (Liu, 2009). Figure 4.7 contains an example output of the software. To speed up the computation of the average flow (Section 4.2.2), we precompute the integral flow of each frame using integral images. More specifically, to compute the sum of the flow vectors in every
pixel of a given box $b$, we simply look up the value of the four corners in the integral flow image, and do the appropriate computation.

Figure 4.6: Typical output of person detection software. Each red box indicates a person detected on the image. Notice the false positive on the right-hand side of the image.

Parameter tuning  
To calibrate relevant parameters of the generative model, we match each detection box to the ground truth box with which it has maximum overlapping area, provided it is greater than 50%, otherwise it is counted as a false detection. Using this matching, we find reasonable values for $\lambda_A$ and for the parameters of the likelihoods $\phi_B$ and $\phi_I$. For the former, we simply average number of detections associated to each ground truth box; we estimate the latter using a maximum likelihood approach (using the ground truth boxes). For $\phi_x$, we use
Figure 4.7: Typical output of optical flow software. Each pixel is color-coded to indicate the direction of the flow. Blue and red indicate left and right, respectively, and yellow and green encode up and down.

an out-of-the-box optimizer to fit the scale and signal variance parameters of the covariance function to the ground truth trajectories.

**Ground truth** We manually annotated boxes for 47 videos from the DARPA Mind’s Eye Year One (ME-Y1) data set, by drawing tight bounding boxes around each target throughout the video. The parameters are all calibrated only on these videos.

2http://www.visint.org/datasets
4.5 Results

We tested our tracker on two widely-used data sets: the *PETS 2009* data set\(^3\), and the *TUD* data set\(^4\). For *PETS* we tested on the S2L1 video, which has over 795 frames, and contains 19 pedestrians walking freely about a very large area. The *TUD* data set contains three videos, called *campus*, *crossing*, and *Stadtmitte*, with 71, 201, and 179 frames, respectively, featuring between 8 and 13 people walking across the screen, and which were taken with a very low camera angle, causing targets to be frequently occluded for long periods of time. Figure 4.8 shows an example quantitative result.

<table>
<thead>
<tr>
<th></th>
<th>Method</th>
<th>MOTA</th>
<th>MOTP</th>
<th>MT</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PETS</strong></td>
<td>Our method</td>
<td>0.83</td>
<td>0.8</td>
<td>0.67</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Roshan Zamir et al. (2012)</td>
<td>0.9</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>Wu et al. (2012)</td>
<td>0.88</td>
<td>×</td>
<td>0.87</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Andriyenko et al. (2012)</td>
<td>0.96</td>
<td>0.78</td>
<td>0.96</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Andriyenko et al. (2011)</td>
<td>0.88</td>
<td>0.76</td>
<td>0.87</td>
<td>0.05</td>
</tr>
<tr>
<td><strong>TUD-X</strong></td>
<td>Our method</td>
<td>0.80</td>
<td>0.78</td>
<td>0.69</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>Roshan Zamir et al. (2012)</td>
<td>0.91</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td><strong>TUD-S</strong></td>
<td>Our method</td>
<td>0.70</td>
<td>0.73</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Roshan Zamir et al. (2012)</td>
<td>0.78</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>Andriyenko et al. (2012)</td>
<td>0.62</td>
<td>0.63</td>
<td>0.67</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Andriyenko and Schindler (2011)</td>
<td>0.60</td>
<td>0.66</td>
<td>0.67</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Andriyenko et al. (2011)</td>
<td>0.68</td>
<td>0.65</td>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td><strong>TUD-C</strong></td>
<td>Our method</td>
<td>0.84</td>
<td>0.81</td>
<td>0.75</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>Yan et al. (2012)</td>
<td>0.85</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of performance of our approach and several state-of-the-art algorithms on the *PETS* and *TUD* (*campus*, *crossing*, and *Stadtmitte*, labeled *TUD-C*, *TUD-X*, *TUD-S*, resp.) data sets using the CLEAR metrics, as well as those proposed by Li et al. (2000b). We report MOTP as normalized distance, and use × for values not reported, or reported in 2D.

\(^3\)http://www.cvg.rdg.ac.uk/PETS2009/a.html
\(^4\)https://www.d2.mpi-inf.mpg.de/node/382
**Performance measures**  We use the CLEAR metrics (Stiefelhagen et al., 2007) which consists of two measurements, multiple object tracking accuracy (MOTA) and multiple object tracking precision (MOTP). MOTA is a measure of false positives, missed targets and track switches, and ranges from $-\infty$ to 1, with 1 being a perfect score. MOTP measures the average distance between true and inferred trajectories, and ranges from 0 to the threshold at which tracks are said to correspond which, as per convention, we set to 1 meter. See Section 2.4 for more details. We also use the evaluation proposed by Li et al. (2000b), of which we are using two metrics: mostly tracked (MT) and mostly lost (ML). We use a threshold of 80% for declaring a target mostly tracked.

4.5.1 Experiments

We ran three sets of experiments. First, we compare our approach with some state-of-the-art trackers on the PETS and TUD data sets. Table 4.1 shows the results of these experiments. We report results on all metrics for our algorithm, and on the published metrics for each algorithm. As we can see, our approach obtains comparable performance across all data sets. It is worth emphasizing that we calibrate our parameters on a completely different data set than that tested here, whereas other algorithms generally do not.

To truly test the generalization power of our tracker, we ran the same experiments, but this time with parameters calibrated on the benchmark videos (PETS and TUD). Table 4.2 has the results for this experiment. As we can see, calibrating on the testing data set provides a small increase in performance. Note that the greatest increase in performance comes in the PETS data set, which is the one that differs the most from the calibration data set.

We also ran experiments designed to test the impact of the different parts of our model, in which we ran our tracker with certain aspects disabled. Here we used the relatively easy TUD-Campus video. The results for these experiments are in Table 4.3. Not surprisingly, the performance took the greatest blow when the tracker ignored optical flow features. These results also suggest that our handling
Table 4.2: Comparison of performance of our approach when calibrating model and inference parameters on our calibration data set and on the benchmark data set. Rows with ME-Y1 indicate calibration on the calibration data set (i.e., the standard way to run our algorithm), while “Benchmark” indicates calibration on the benchmark data sets: PETS and TUD (campus, crossing, and Stadtmitte, labeled TUD-C, TUD-X, TUD-S, resp.) We use the CLEAR metrics, as well as those proposed by Li et al. (2000b), and report MOTP as normalized distance.

Table 4.3: A summary of the effect of removing key features of our tracker. “Base” is our full algorithm, “NO-OF” ignores optical flow features, NO-OCC does not reason about occlusion, and “NO-SZ” does not estimate person size. We use the CLEAR metrics (MOTA and MOTP), as well as those proposed by Li et al. (2000b) (MT and ML). We report MOTP as normalized distance.

4.6 Discussion

In this chapter, we presented another application of our temporal scene understanding approach, this time to videos of people walking in a 3D scene. Our main goal was to show that our approach is applicable to different kinds of scenarios; indeed,
Figure 4.8: Visualization of some of our results: four frames of the *PETS-S2L1* video with the 3D scene super-imposed.

The videos used here are completely different from those used in Chapter 3. Both the scene and the imaging system were more complex, each of which presented its own challenges.

The complexities in the scene included a more sophisticated model for targets, i.e., a 3D cylinder instead of a point. This means we must infer additional variables, which is always a challenge. The non-linear camera transformation also posed a significant challenge, in that it forced us to approximate a very difficult integral. As the results show, our approach overcomes both of these difficulties well, and even achieves performance comparable to algorithms which are tailored to this problem (and, sometimes, to these specific data sets). The results also show that our Gaussian process motion model is very general, as the motion exhibited by humans is quite different from that of growing pollen tubes, and all they have in common is that they can be reasonably described as “smooth”.
Nonetheless, our algorithm has several shortcomings, which can be identified by careful examination of the qualitative results. First, the model for people is much too simple. This causes several issues. For instance, the model box projected onto the image plane contains too many pixels which are not part of the person’s projection. This reduces the accuracy and usefulness of the image likelihood, as it relies on the pixel flow vectors as cues. We address this in the next chapter, by adding several additional parameters to the 3D person model. In addition to addressing a practical issue, a more complete person model constitutes a better understanding of the scene, which is, after all, the goal of our approach.

Another current drawback of our approach is its time complexity. Currently, it takes approximately 10 seconds of computation time per frame on a modern desktop computer. Although we perform optimizations and approximations which significantly reduce the runtime, we have not yet fundamentally addressed this issue. However, we note that our algorithm is easily parallelizable, making it possible to speed up runtime significantly.
CHAPTER 5

TRACKING PEOPLE’S GAZE IN 3D

5.1 Introduction

In this chapter, we explore another application of our scene understanding approach to a similar problem. Specifically, we wish to expand our understanding of scenes with people to include their gaze direction. To do this, we work off the application presented in Chapter 4, and extend the person model to include the head of people, which we represent with a 3D ellipsoid. This new model has many applications, including surveillance and automated driving systems.

This problem presents several additional challenges when compared to that of simply tracking position and size. Perhaps the most difficult issue to overcome is that obtaining evidence for gaze direction from videos is very difficult. Indeed, this area is still in its infancy, especially when compared to object detection, which is very mature.

In the following sections, we will examine the components of our model as applied to this problem. We model people as upright 3D cylinders moving on the ground plane, whose motion is described by a Gaussian process. Additionally, we model people’s heads as a 3D ellipsoid, which sits atop the cylinder, which represents the body. The cylinder has two degrees of rotational freedom, pitch and yaw. We use the perspective camera representation to project the scene onto the image plane, and model detections as noisy bounding boxes around the projected cylinders and ellipsoids, and optical flow features as noisy displacement vectors of consecutive cylinder projections. For inference, we use the approach detailed in Section 2.3. We will also discuss our method for approximating the integral over scenes, which, due to the perspective projection and non-Gaussian likelihood, does not have closed form.
We test our method on benchmark videos, and compare our performance with some of the current state-of-the-art trackers. Our experiments show performance which is comparable to other trackers using standard evaluation measures. It is worth noting that we estimate the camera and the size of each person, whereas other trackers do not.

5.1.1 Related work

Head pose estimation has been the subject of study for many years, and many approaches have been proposed to solve the problem. Appearance-based models, such as Beymer (1994) and Niyogi and Freeman (1996), determine pose by finding the most similar image amongst a set of pre-built templates. There are also approaches based on face detection, which train several detectors, each on a different (discrete) pose. Huang et al. (1998) trained detectors for three different yaws using support vector machines. Nonlinear regression methods estimate pose by learning a non-linear function from the image space to one or more pose directions. In this vein, PCA-based methods have tried with some success (Li et al., 2000a, 2004).

There are also geometric methods, which use head shape and precise configuration of local features to estimate pose. For example, given five facial points (the outside corners of the eyes, the outside corners of the mouth, and the tip of the nose), the axis of facial symmetry can be found by connecting a line between the midpoint of the eyes and the midpoint of the mouth (Gee and Cipolla, 1994; Horprasert et al., 1997).

Head pose estimation can also be done in the context of tracking. Bottom-up approaches follow low-level facial landmarks from frame to frame. Yao et al. (2001) assume that the human face is planar, and recovers the affine transformation between frames using least-squares. More recently, Ohayon and Rivlin (2006) uses SIFT descriptors and use prior knowledge about the 3D face shape to track the head pose. There has also been work in using a 3D rigid head model, and finding the rotation and translation of the model that best fits each new image observation (Pappu and Beardsley, 1998; Schodl et al., 1998; Malciu and Prêteux, 2000).
5.2 Model

We use the Bayesian generative formulation discussed in Chapters 2 and 4. We have three sources of data, person detections, optical flow features, and facial landmark detections (see Section 5.4 for details on how we obtain it), which we assume to be the result of the process implied by our generative model. First, an association object $\omega$ is sampled from the prior $p(\omega)$. Then, the 3D scene $z$ and camera $C$ are generated from $p(z)$ and $p(C)$, respectively. Each person in the scene is projected onto the image plane via the camera, resulting in a set of bounding boxes, displacement vectors, and 2D face features, which are perturbed to generate our data detections $B$, and flow features and facial landmarks $I$, according to the likelihood function $f(B, I | \omega, z, C)$. We examine these in detail in the following sections. We exclude from the discussion items described in previous sections, such as the model for data association (discussed in Section 2.2.1) and the camera model (discussed in Section 4.2.1).

5.2.1 Scene and camera

We model the scene as a set of people in 3D. The scene $z$ consists of each individual person’s state $z_r$, $r = 1, \ldots, m$. In turn, each person’s state can be decomposed into their trajectory $x_r = (x_{r1}, \ldots, x_{rl})$ and their physical state, which is composed of their height, width, and girth, and the pitch and yaw of their head, i.e., $w_r = (h_r, w_r, g_r, p_r, y_r)$. We can think of each person as being represented by a $h_r \times w_r \times g_r$ cylinder with an oriented ellipsoid on top of it that moves on the ground plane. We assume the trajectory of each person is a realization of a Gaussian process with a kernel which promotes smooth motion. Under this model, we have that

$$p(x_r) = \prod_{i=1}^{D_x} p(x^i_r),$$

(5.1)

where $x_i$ is the vector of the $i$th components of $x_r$, and $D_x = 2$, since the ground plane is 2D. Each individual prior $p(x^i_r)$ is the Gaussian pdf which results from the Gaussian process. See Section 2.2.2 for more details. Person size is a priori normally
distributed, as described in Section 4.2.1, given by equations 4.2, 4.3, and 4.4. The head orientation also has a Gaussian process prior, where each angle is assumed to be independent of the other. Following the same model as that of position, we have that the prior on head pitch is given by

\[ p_r \sim \mathcal{N}(0, K_r) \]  

(5.2)

and the prior over head yaw by

\[ y_r \sim \mathcal{N}(0, K_r), \]  

(5.3)

where \( p_r = (p_{r1}, \ldots, p_{rl_r}) \) and \( y_r = (y_{r1}, \ldots, y_{rl_r}) \). The prior over a person’s full physical state is \( p(w_r) = p(h_r)p(w)p(g_r)p(p_r)p(y_r) \), since we assume all variables to be independent of each other. Figure 5.1 illustrates the person model.

![Figure 5.1: Each person is modeled by an upright cylinder with a ellipsoid sitting atop of it, whose bottom is at position \( x_{rj} \). The cylinder has three dimensions – height (shown as \( h_r \)), width, and girth – and the head has orientation given by two angles: its pitch \( p_{rj} \) and its yaw \( y_{rj} \).](image)

As before, the prior over a person’s state is \( p(z_r) = p(x_r)p(w_r) \), and the full
The scene prior is given by

\[
p(z | \phi_z, \phi_w) = \prod_{r=1}^{m} p(z_r | \phi_x, \phi_w),
\]

(5.4)

The parameters \( \phi_x \) are set from data (see Section 4.4), and \( \phi_w = (\mu_h, \sigma_h, \mu_w, \sigma_w, \mu_g, \sigma_g) \) are set manually, chosen to follow actual human size, whose distribution is detailed by McDowell et al. (2005). The camera model and prior is exactly the same as the one previously used, so it will not be discussed here. See Section 4.2.1 for details.

**Scene projection**

We project the scene in a very similar way as before. For each person, at each frame, we obtain a model box in the same way as before (see Figure 4.1). However, for each person \( r \) in its \( j \)th frame we now compute two additional boxes. First, we compute a bounding box around the projection of the head ellipsoid, which we call the model face box. Second, we get find a bounding box around the reduced cylinder in a very similar way, which we call the model body box. We denote these two boxes by \( f_{rj} \) and \( o_{rj} \), respectively. See Figure 5.2 for an illustration. Further, occlusion reasoning becomes more precise as a result of this new computation. The visible region of person \( r \) at frame \( j \), \( \hat{h}_{rt} \), is computed in a similar way as before, but using \( f_{rj} \) and \( o_{rj} \) for the computation, instead of \( h_{rj} \).

We also redefine the concept of model vector, and we distinguish between the model face vector and the model body vector. To compute the model face vector for person \( r \) at its \( j \)th frame, we first pick a visible point on the 3D head ellipsoid at frame \( j \). We then project that point onto the image in frames \( j \) and \( j + 1 \), and the model face vector \( c_{rj} \) is given by the difference between the two projected points. We perform the analogous computation using the body cylinder to get the model body vector \( u_{rj} \). Performing the computation in this way has several advantages compared to the previous way of doing it (i.e., Section 4.2.1). First, it adds granularity to the model flow, which allows the head pixels to move in a different direction than the body pixels. Second, choosing the points in 3D prior to projecting them allows us
Figure 5.2: For person $r$, at its $j$th frame, we find three model boxes. The original model box $h_{rj}$ is a bounding box around the whole person. The model body box $o_{rj}$ surrounds the body cylinder. Finally, the model face box $f_{rj}$ is computed around the projection of the head ellipsoid.

to model rotation without motion. In the previous way of doing it, only flow as a consequence of rigid motion could be captured.

Finally, we also project the predicted facial features of each person. Given the head ellipsoid of person $z_r$ at its $j$th frame and its orientation $(p_{rj}, y_{rj})$, we know where the facial features are in 3D, which we can project onto the image using the camera $C$. From this, we seven facial features $m_{rj} = (m_{rj}^1, \ldots, m_{rj}^7)$, the numbering of which corresponds to that of the detected landmarks $(k^1, \ldots, k^7)$. 
5.2.2 Likelihood

We use three sources of evidence: person detectors, optical flow, and face landmarks. First, we run various person detectors on the video frames to get bounding boxes $B_t = \{b_{t1}, \ldots, b_{tN_t}\}$, $t = 1, \ldots, T$, where $N_t$ is the number of detections in frame $t$. We parametrize each box $b_{tj}$ by $(b_{txj}, b_{tyj}, b_{tuj})$, representing the $x$-coordinate of the center, and the $y$-coordinates of the top, and bottom, respectively. We also run a dense optical flow estimator on the video, which outputs a set of velocity vectors $I^f_t = \{v_{t1}, \ldots, v_{tN_t}\}$ for each frame $t = 1, \ldots, T - 1$, where $N_t$ is the number of pixels in the frame. Finally, we run a face landmark detection, which provides seven 2D points for each face, $k_{ti} = (k_{t1i}, \ldots, k_{t7i})$, representing the corners of the eyes and mouth, as well as the tip of the nose, of the $i$th detection at frame $t$. We use $I^k_t = \{k_{t1}, \ldots, k_{tN_t}\}$ to represent all face landmarks detected at frame $t$, and $I^k = \{I^k_1, \ldots, I^k_T\}$. As before, we use $B = \bigcup_{t=1}^T B_t$ and $I = (I^f, I^k)$. We use the exact same detection likelihood as in the previous chapter; consequently, we shall not discuss it here.

**Image likelihood**

The image likelihood will now consist of two separate parts: the optical flow likelihood, and the face landmark likelihood. The optical flow component is very similar to the one described previously, but we extend it in the natural way, by having two independent factors, one for each model vector, and each analogous to the whole model box likelihood described in Section 4.2.2. That is, for the model body vector $o_{rj}$, its likelihood is a Laplace pdf, i.e., $v^x \mid u_{rj}^x \sim \text{Laplace}(u_{rj}^x, \sigma_{rj}^x)$ and $v^y \mid u_{rj}^y \sim \text{Laplace}(u_{rj}^y, \sigma_{rj}^y)$. Similarly, for the model face vector, we have $v^x \mid c_{rj}^x \sim \text{Laplace}(c_{rj}^x, \sigma_{rj}^x)$ and $v^y \mid c_{rj}^y \sim \text{Laplace}(c_{rj}^y, \sigma_{rj}^y)$ (see Figure 5.3). In this way, we get an optical flow likelihood given by

$$p(I^f \mid z, \omega, C) \propto \prod_{t=1}^{T-1} \prod_{v \in I^f_t} \frac{p(v \mid u(v), C, \phi_{II})}{p(v \mid \phi_{II})}.$$  \hspace{1cm} (5.5)

See Section 4.2.2 for its derivation.
Figure 5.3: The optical flow likelihood. Here, model body direction $u_{rj}$ (blue arrow) is compared against $v$ (red arrow), the average of all the flow vectors inside $\hat{o}_{rj}$, which only includes the visible part of $o_{rj}$ (big blue box), since it is partially occluded by the red model boxes, presumably belonging to a different person. Similarly, model face direction $c_{rj}$ is compared against $v'$, the average of all flow vectors inside $\hat{f}_{rj}$ (top blue box). Again, we exclude flow vectors in the red model face box of the second (red) person.

For the face landmark, we simply assume a Gaussian noise model around each of the model face features $m_{rj}$. In other words, we have that, for every $k \in I^k$

$$
k^i \sim \mathcal{N}(m^i_{rj}, \Sigma^k_{I^k})
$$

(5.6)

for $i = 1, \ldots, 7$, where $m^i_{rj}$ is the corresponding model face feature to $k^i$. See Figure 5.4 for an illustration of the likelihood. Assuming independence of all landmarks, we get a landmark likelihood of

$$p(I^k | z, \omega, C) = \prod_{m \in I^k} p(m | k(m), \phi_{I^k}),$$

(5.7)

where $k(m)$ is the predicted face feature corresponding to landmark $k$, and $\phi_{I^k} = \Sigma_{I^k}$. 

Figure 5.4: The face landmark likelihood. The blue box represents the model face box $c_{rj}$, the blue circles are the projected face features $m_{rj}^1, \ldots, m_{rj}^7$, and the red $\times$s are the corresponding landmark detections $k^1, \ldots, k^7$. The landmark detections are independently normally distributed around their corresponding face feature.

Then, the full image likelihood is simply given by

$$p(I \mid z, \omega, C) = p(I^f \mid z, \omega, C)p(I^k \mid z, \omega, C).$$

(5.8)

5.3 Inference

We use the same inference process as that of the previous chapter, with only two minor differences. First, the vector scene vector $z$ is now significantly higher-dimensional. However, our inference approach is agnostic to this, and requires no change. The other difference is in how we obtain the initial scene $z^{(0)}$ for sampling. In addition to obtaining the initial trajectories $x^{(0)}$ and person sizes, we must also initialize the head orientations. For each person $r$ at each of their frames $j$, we compute the motion direction, which is given by difference between consecutive positions. That is, we set $y_{rj}^{(0)} = x_{rj+1}^{(0)} - x_{rj}^{(0)}$. Finally, we simply set the pitch to
zero, i.e., $p_{rj}^{(0)} = 0$. In other words, we set the initial head orientation to be looking straight ahead in the same direction one is walking.

5.4 Data, tuning, and ground truth

As mentioned previously, we obtain the detection boxes and optical flow vectors using freely-available software. Section 4.4 has the details regarding this, and the preprocessing of this data. The face landmark data is obtained using the detector developed by Zhu and Ramanan (2012), which we run independently on each frame of a video. Figure 5.5 has an example output of this detector. The calibration of relevant parameters is also described in the previous chapter. The parameters of the landmark likelihood $\phi_{rk}$ are tuned manually for this scenario.

![Figure 5.5: Typical output of landmark detection software.](image_url)

**Ground truth** Unfortunately, none of the benchmark videos we use feature clearly-visible face features of the actors. Consequently, we introduce our own data set, for which we annotate the head orientation as follows. In the data set, the actors
were instructed to constantly look at a laser-pointer which moves about the ground plane and on other actors in the scene. We then manually annotate the positions of each person and of the laser pointers. Given the inferred ground plane and the ground truth positions of every person, we can estimate the head orientation by shooting a ray from the head of a person to the position of the laser pointer on the ground. This data set contains two videos, each featuring three people in an indoor scene. In the first video, named *UA-standing*, each person is standing still, while their gaze is following the laser pointer. In the second video, named *UA-walking*, the same thing happens, except that each person moves freely around the room as they follow the laser pointer with their gaze.

5.5 Results

We test our algorithm in two different ways, by answering two questions. First, we wish to understand if the more sophisticated person model improves the results obtained using the simpler model. To do so, we run our approach on the two data sets introduced previously, *TUD* and *PETS*. The results are shown in Table 5.1. We can see from the results that the extension to the person model indeed improves performance, except in the case of *PETS*, which can be explained by the fact that this particular data set has very low resolution, and the camera is quite far away from the actors.

We also want to test how well the head orientation is estimated. For this purpose, we use a modified MOTP measure, where we use distance between angles, measured in degrees, instead of between feet position. That is, for any pair of ground-truth and estimated trajectories which are determined to correspond (see Section 2.4), we compute the average error in the angle between the ground truth and estimated gaze directions. We measure this on the *UA* data set introduced in Section 5.4. Figure 5.6 shows an frame with example output. We also report performance on standard MOTA and MOTP. The results are shown in Table 5.2. Although the performance looks promising, it is difficult to ascertain the quality of these results, since there
Table 5.1: Comparison of performance of our approach when using the base person model (labeled “Base”) with that of using the more complete model described in this chapter (“Base+Head”). We test on the two datasets discussed previously: PETS and TUD (campus, crossing, and Stadtmitte, labeled TUD-C, TUD-X, TUD-S, resp.) We use the CLEAR metrics reporting MOTP as normalized distance.

<table>
<thead>
<tr>
<th>Method</th>
<th>MOTA</th>
<th>MOTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>0.83</td>
<td>0.8</td>
</tr>
<tr>
<td>Base+Head</td>
<td>0.83</td>
<td>0.82</td>
</tr>
<tr>
<td>Base</td>
<td>0.80</td>
<td>0.78</td>
</tr>
<tr>
<td>Base+Head</td>
<td>0.81</td>
<td>0.85</td>
</tr>
<tr>
<td>Base</td>
<td>0.70</td>
<td>0.73</td>
</tr>
<tr>
<td>Base+Head</td>
<td>0.73</td>
<td>0.71</td>
</tr>
<tr>
<td>Base</td>
<td>0.84</td>
<td>0.81</td>
</tr>
<tr>
<td>Base+Head</td>
<td>0.85</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Table 5.2: Performance of our approach on the UA data set, introduced by the authors. MOTA and MOTP are the standard CLEAR metrics, while MOTP-A is the average error in angle between corresponding annotated and estimated head orientation.

<table>
<thead>
<tr>
<th>Video</th>
<th>MOTA</th>
<th>MOTP</th>
<th>MOTP-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>UA-standing</td>
<td>0.98</td>
<td>0.91</td>
<td>18°</td>
</tr>
<tr>
<td>UA-walking</td>
<td>0.79</td>
<td>0.81</td>
<td>35°</td>
</tr>
</tbody>
</table>

5.6 Discussion

In this chapter, we presented a third application of our temporal scene understanding approach, where we extend our person model to include a head and its orientation. We test our approach on benchmark data sets, and show that using the extended person model improves the already state-of-the-art performance. We also show initial results on tracking gaze direction, using the same Bayesian paradigm. For this purpose, we introduce a new dataset which we plan to make publicly available.
Figure 5.6: A single frame of the *UA-walking* video, with the result super-imposed on the image. The large boxes are the estimated body boxes and the small boxes are the head boxes. The dots inside the small boxes are the predicted locations of the face features: the eyes, the nose, and the corners of the mouth.

The results further demonstrate the power of our Bayesian approach, as it can be applied to different kinds of scenarios with good performance. With this in mind, future work includes further extensions to the person model, with the goal of understanding the scene in a more complete way.
In this chapter, we discuss some of the details of the implementation of our approach. We note that knowledge of the details discussed here is *not* necessary for obtaining an understanding of the material presented in this dissertation.

A.1 Software

All algorithms presented in this work were implemented in the C++ programming language, compiled with the GNU Compiler Collection\(^1\) versions 4.5 through 4.7. Besides the standard C and C++ libraries, we made use of numerous software tools. Most notably, we worked with the KJB library, which is an internal C/C++ library maintained by the members of our research group. We also made heavy use of *Boost*, which is a set of C++ libraries which provide support for many different tasks and structures, such as random number generation and I/O management. Another software tool upon which we depend is the *OpenGL* API\(^2\), which we use to visualize our 3D scenes. We also heavily depend on the *LAPACK*\(^3\) interface, which is a library that provides an efficient implementation of many linear algebra operations and algorithms. Finally, all sampling algorithms used are from *ergo*\(^4\), an MCMC sampling library which is partly developed and maintained by the author.

A.2 Efficient gradient computation

As mentioned in Section 4.3, the HMC algorithm requires the gradient computation when simulating dynamics. Specifically, we need to compute the gradient of

\(^{1}\)http://gcc.gnu.org
\(^{2}\)http://www.opengl.org
\(^{3}\)http://www.netlib.org/lapack
\(^{4}\)https://github.com/ksimek/libergo
the scene posterior (Eq. 4.33), which is very difficult to do analytically, due to the complexity of the likelihood function. Instead, we recur to numerical differentiation, in the form of central finite differences (CFD), which gives us the following approximation for the ith component of the gradient vector:

$$\frac{\partial}{\partial z_i} \log p(z \mid B, I, \omega, C) \approx \frac{\log p(z + \varepsilon_i e_i \mid B, I, \omega, C) - \log p(z - \varepsilon_i e_i \mid B, I, \omega, C)}{2\varepsilon_i}$$

(A.1)

where \(\varepsilon_i\) is the step size for the ith dimension, and \(e_i\) is the ith standard basis (unit) vector. When done naively, the computation of the whole gradient vector requires \(2D_z\) evaluations of the posterior (Eq. 4.33), which is expensive, especially considering that \(D_z\) is typically very large.

We speed up this computation significantly by exploiting the conditional independence between many of the variables in our model. First, note that the single-detection likelihood (Eq. 4.15) only has the model box of the target to which it corresponds to on the conditioning side. This means that we can rewrite Eq. 4.17 as

$$p(B \mid z, \omega, C) = \prod_{b \in \tau_0} p(b) \prod_{t=1}^{l_r} \prod_{j=1}^{l_{r_{t,j}}} p(\tau_{r_{t,j}} \mid h_{r_{t,j}})$$

(A.2)

where \(\tau_{r_{t,j}}\) is the detection box corresponding to person \(r\) in its \(j\)th frame. (We are abusing notation here, since targets need not have an associated detection in all frames in which they exist.) Then, noting that \(h_{r_{t,j}}\) only depends on \(x_{r_{t,j}}\), and taking the log, we get

$$\log p(B \mid z, \omega, C) = \sum_{b \in \tau_0} \log p(b) \sum_{r=1}^{m} \sum_{j=1}^{l_{r_{t,j}}} \log p(\tau_{r_{t,j}} \mid x_{r_{t,j}}),$$

(A.3)

Following the same reasoning, we can rewrite the image log-likelihood (Eq. 4.20) as

$$\log p(I \mid z, \omega, C) = \text{const} + \sum_{r=1}^{m} \sum_{j=1}^{l_{r_{t,j}-1}} (\log p(v_{r_{t,j}} \mid x_{r_{t,j}}, x_{r_{t,j}+1}) - \log p(v_{r_{t,j}}))$$

(A.4)

where \(v_{r_{t,j}} \in I^*\) is the average flow vector corresponding to model vector \(u_{r_{t,j}}\). Here, we have used the fact that \(u_{r_{t,j}}\) is a function of \(x_{r_{t,j}}\) and \(x_{r_{t,j}+1}\). We can now define the
likelihood “of a single model variable”, which is the factors of the likelihood which affect only a single scene variable:

\[ L^i(z) = \log p(\tau_{rj} \mid x_{rj}) + \log p(v_{rj-1} \mid x_{rj-1}, x_{rj}) + \log p(v_{rj} \mid x_{rj}, x_{rj+1}), \quad (A.5) \]

where \( x_{rj} \) is the scene variable corresponding the \( i \)th dimension of \( z \). Clearly, this is a much faster computation than the full likelihood (Eq. 2.18). If we substitute equations A.3 and A.4 into A.1, it is not too difficult to see that we get

\[
\frac{L^i(z + \varepsilon_i e_i) + \log p(z + \varepsilon_i e_i) - L^i(z - \varepsilon_i e_i) - \log p(z - \varepsilon_i e_i)}{2\varepsilon_i}.
\]

This re-factorization of the gradient significantly speeds-up the computation, noticeably reducing the run-time of the whole system.

**Hessian computation**  As detailed in Section 4.3, we approximate the likelihood in Eq. 4.26 using the Laplace-Metropolis approximation, given by Eq. 4.32, which requires the computation of the Hessian matrix of the negative log-posterior over scenes, i.e.,

\[ -H = \frac{\partial^2}{\partial z^2} \log p(z \mid B, I, \omega, C). \quad (A.7) \]

As with the gradient vector, an analytical derivation of this matrix is impossible, due to the complexity of the likelihood function, and we resort to CFD to approximate its elements. That is, element \( i, j \) of the matrix can be approximated as

\[
-H_{ij} = \frac{\partial^2}{\partial z_i \partial z_j} \log p(z \mid B, I, \omega, C) \approx \frac{1}{4\varepsilon^2_i \varepsilon^2_j} \left( \log p(z + \varepsilon_i e_i + \varepsilon_j e_j \mid B, I, \omega, C) \right.
\]

\[
- \log p(z + \varepsilon_i e_i - \varepsilon_j e_j \mid B, I, \omega, C)
\]

\[
- \log p(z - \varepsilon_i e_i + \varepsilon_j e_j \mid B, I, \omega, C)
\]

\[
+ \log p(z - \varepsilon_i e_i - \varepsilon_j e_j \mid B, I, \omega, C) \right) .
\]

Following the same reasoning as we did for the gradient, we can obtain a significant speed-up for this computation. However, since the computation of \( H \) requires \( O(D^2) \)
evaluations, calculating all of its elements remains intractable. To address this, we must approximate further, which we do in the following way. First, note that most of the elements of $H$ are very close to 0, since most of the variables of $z$ are practically uncorrelated, a fact that we can exploit by only computing a small subset of them. More specifically, we only compute the $H_{ij}$ if we expect $z_i$ and $z_j$ to covary, e.g., when $i = j$. Using this approximation, we get an important reduction in run-time, making the approximation of $H$ tractable.
In this appendix, we give a brief review of Markov chain Monte Carlo methods, focusing on the algorithms used in this work. This is not intended to be comprehensive analysis; please refer to the cited material for a complete examination of the material covered here.

B.1 Background

Markov chain Monte Carlo (MCMC) is a class of algorithms for sampling from a probability distribution, based on creating a Markov chain which has the target distribution as its stationary distribution. Such algorithms have many applications in Bayesian statistics, such as approximating density functions, estimating integrals, and finding the maximum \textit{a posteriori} values. In our particular application we use three different algorithms, Metropolis-Hastings, Gibbs, and Hamiltonian Monte Carlo, to maximize different pdfs. In this chapter, we shall discuss a few of the details of these algorithms in a general context. In what follows, the goal will be to generate samples according to distribution $\pi(x)$, $x \in \mathcal{X}$, which we call the \textit{target} distribution. We also use $\pi$ to denote the pdf of the distribution.

B.2 Metropolis-Hastings

The Metropolis-Hastings (MH) algorithm works by building a Markov chain which has $\pi$ as its stationary distribution. In other words, we want a Markov chain whose marginal distribution is $\pi$ at all times, regardless of its initial state, so that simulating the chain gives us samples from $\pi$, which is our ultimate goal.

Markov chain $x^{(1)}, x^{(2)}, \ldots$ is determined by the distribution of its initial state $x^{(1)}$, and a transition density, $p(\cdot | \cdot)$, which specifies the conditional distribution of
state $x^{(i)}$, given the previous state $x^{(i-1)}$. In other words, from current state $x^{(i)}$, the probability that $x^{(i+1)}$ is in some set $A \in \mathcal{X}$ is given by

$$
\int_A p(x^{(i+1)} | x^{(i)}) \, dx^{(i+1)}. 
$$

(B.1)

The distribution $\pi$ is invariant to a Markov chain if the transition distribution $p$ preserves $\pi$, i.e., if $x^{(i)} \sim \pi$ implies $x^{(i+1)} \sim \pi$. Put another way, $\pi$ is invariant if

$$
\int_B \int_A p(x' | x) \pi(x) \, dx' \, dx = \int_B \pi(x) \, dx,
$$

for any $B \subset \mathcal{X}$. Showing invariance can be difficult, and, instead, we impose the stronger constraint of reversibility. A Markov chain is reversible if

$$
\int_B \int_A p(x' | x) \pi(x) \, dx' \, dx = \int_A \int_B p(x' | x) \pi(x) \, dx' \, dx
$$

for any $A, B \subset \mathcal{X}$. It is straightforward to show that reversibility implies invariance.

MH gives us a recipe for building a reversible chain in a simple way. Let $q(x | \cdot)$ be a conditional distribution with support $\mathcal{X}$, called the proposal distribution. In addition, let $x^{(i-1)}$ be the current state (sample). At each iteration $i$ of the algorithm, we generate sample $x^{(i)}$ by first drawing a proposal sample from $q$, $x' \sim q(\cdot | x^{(i-1)})$, and then accepting it or rejecting with probability

$$
\max \left( 1, \frac{\pi(x') q(x^{(i-1)} | x')}{\pi(x^{(i-1)}) q(x' | x^{(i-1)})} \right),
$$

(B.4)

where the ratio in the max is called the acceptance ratio or MH ratio. If we accept, then $x^{(i)} = x'$; otherwise, $x^{(i)} = x^{(i-1)}$. Figure B.1 shows the MH algorithm. It can be shown that a Markov chain with this transition distribution is reversible and, therefore, has invariant distribution $\pi$. It is worth noting that, although MH has very nice theoretical properties, in practice, the number of samples necessary for the chain to mix (for $\pi$ to become the invariant distribution of the chain) could be prohibitively large. Not surprisingly, the choice of proposal distribution $q$ turns out to play a crucial role in the performance of the sampler. There has been much work done in this area (see Section 1.1), but the best results come from proposal distributions which are largely application-specific, especially for multi-modal target distributions in spaces of high dimensionality. For more details, we refer the reader to the work of Neal (1993).
### Algorithm B.1: The Metropolis-Hastings algorithm

**Input:** $x^{(0)}$, $\pi$, $q$, $N$

for $i = 1, \ldots, N$ do

- Draw $x' \sim q(\cdot | x^{(i-1)})$
- Set $p_{\text{accept}}$ to the acceptance probability $\triangleright$ (eq. B.4)
- Draw $u \sim U(0,1)$
- if $u < p_{\text{accept}}$ then
  - Set $x^{(i)} = x'$
- else
  - Set $x^{(i)} = x^{(i-1)}$

Figure B.1: The Metropolis-Hastings algorithm. Starting with initial state $x^{(0)}$, MH draws $N$ samples from $\pi$ using proposal distribution $q$. See Section B.2 for details.

### B.3 Gibbs

The Gibbs algorithm is a special case of Metropolis-Hastings. Assume that we can draw samples from the full conditional distributions $\pi_j(x_j | x_{-j})$, where $x_{-j}$ is vector $x$ with variable $x_j$ removed. If we let the proposal distribution in MH be this full conditional distribution, and plug it into Eq. B.4, it is easy to see that we get an acceptance ratio of 1, with all factors canceling. The Gibbs sampler takes advantage of this, which makes for a very simple algorithm. At iteration $i$ of Gibbs, we successively draw samples from $\pi(x_j' | x_{-j}^{(i-1)})$. Then, $x^{(i)} = (x_1', \ldots, x_N')$ is a sample from the target distribution $\pi$. Figure B.2 contains the pseudocode for the algorithm.

### B.4 Hamiltonian Monte Carlo

Another special case of MH is the Hamiltonian Monte Carlo (HMC) algorithm, also known as the Hybrid Monte Carlo, developed by Radford Neal. In this algorithm, we shall build a Markov chain in a similar way to MH, but we will use the gradient of the target distribution in order to explore the space more efficiently. Indeed, one of the main advantages of HMC over MH is that HMC does not exhibit random-walk behavior, which severely limits algorithms like MH and Gibbs.
Algorithm B.2: The Gibbs algorithm

Input: \(x^{(0)}, \pi, N\)

for \(i = 1, \ldots, N\) do
  Let \(x^{(i)} = x^{(i-1)}\)
  for \(j = 1, \ldots, D_x\) do
    Draw \(x' \sim \pi_j(\cdot | x^{(i)}_{-j})\)
    Let \(x^{(i)}_j = x'\)

Figure B.2: The Gibbs algorithm. Starting with initial state \(x^{(0)}\), Gibbs draws \(N\) samples from \(\pi\) by successively sampling from the full-conditional distributions \(\pi_j(x_j | x_{-j})\). See Section B.3 for details.

HMC simulates a particle moving in a physical system, where \(x\) is its position, and \(U(x) = -\log \pi(x)\) is the potential energy. We introduce another variable \(p\) to represent the momentum, and \(K(p) = p^\top p\) is the kinetic energy of the system (i.e., \(p\) is normally-distributed). Given current sample \(x^{(i-1)}\), we obtain the next sample in the following way. First, we draw an initial momentum \(p \sim \mathcal{N}(0, I)\). We will then evolve Hamiltonian dynamics for \(L\) steps using the leapfrog method (see below), with a step size of \(\varepsilon\), resulting in the proposed joint state \((x', p')\), which we accept with probability

\[
\min \left(1, \frac{\pi(x') f(p')}{\pi(x) f(p)}\right),
\]

where \(f(p) = e^{-K(p)}\). As before, if we accept the sample, \(x^{(i)} = x'\), and \(x^{(i)} = x^{(i-1)}\) otherwise. Assuming dynamics is evolved correctly (i.e., the method used preserves volume), it can be shown that samples \(x^{(1)}, x^{(2)}, \ldots\) generated in this way are distributed according to \(\pi\). Figure B.3 shows the pseudocode for the HMC algorithm.

Simulating Hamiltonian dynamics is not trivial, because we must do it in discrete time steps. That is, given position and momentum at time \(t\), \(z(t)\) and \(p(t)\), we need to calculate their values at time \(t + \varepsilon\), \(z(t + \varepsilon)\) and \(p(t + \varepsilon)\), while preserving the volume of the system. There are several algorithms which can be used for this purpose. We use the leapfrog method, which consists of three steps. First, we perform a half-step update of the momentum, followed by a full-step update of the
Algorithm B.3: The Hamiltonian Monte Carlo algorithm

**Input:** $x^{(0)}$, $\pi$, $\epsilon$, $L$, $N$

**for** $i = 1, \ldots, N$ **do**

Draw $p \sim \mathcal{N}(0, I)$ \Comment{(initial momentum)}

Let $x' = x^{(i-1)}$

Let $p' = p$

**for** $t = 1, \ldots, L$ **do** \Comment{(simulate dynamics)}

Let $p' = p' - \frac{\epsilon}{2} \nabla U(x')$

Let $x' = x' + \epsilon p$

Let $p' = p' - \frac{\epsilon}{2} \nabla U(x')$ \Comment{second half-step momentum update}

Set $\alpha$ to acceptance probability \Comment{(eq. B.5)}

With probability $\alpha$, accept: $x^{(i)} = x'$

With probability $1 - \alpha$, reject: $x^{(i)} = x^{(i-1)}$

---

Figure B.3: The Hamiltonian Monte Carlo algorithm. Given initial state $x^{(0)}$, target distribution $\pi$ (where $U = -\log \pi$), and step size $\epsilon$, HMC produces $N$ samples from the $\pi$, using the leapfrog method for dynamics with $L$ steps. See text for some details.

Position; finally, the final half-step update of the momentum is computed. This gives the following update rules:

$$p \left( t + \frac{\epsilon}{2} \right) = p(t) - \frac{\epsilon}{2} \nabla U(x(t)) \quad \text{(B.6)}$$

$$x(t + \epsilon) = x(t) + \epsilon p \left( t + \frac{\epsilon}{2} \right) \quad \text{(B.7)}$$

$$p(t + \epsilon) = p(t + \epsilon) - \frac{\epsilon}{2} \nabla U(x(t + \epsilon)) \quad \text{(B.8)}$$

where $\nabla U(x(t))$ is the gradient (vector) of $U$ with respect to $x$, evaluated at $x(t)$. For a more in-depth discussion of the HMC algorithm, see Neal (2011).
REFERENCES


