Iterative Local Model Selection for Tracking and Mapping

Aleksandr V. Segal
St Anne's College

Active Vision Group
Department of Engineering Science
University of Oxford

Michaelmas Term 2014

This thesis is submitted to the Department of Engineering Science, University of Oxford, for the degree of Doctor of Philosophy. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Aleksandr V. Segal
St Anne's College

Doctor of Philosophy
Michaelmas Term 2014

Iterative Local Model Selection for Tracking and Mapping

Abstract

The past decade has seen great progress in research on large scale mapping and perception in static environments. Real world perception requires handling uncertain situations with multiple possible interpretations: e.g. changing appearances, dynamic objects, and varying motion models. These aspects of perception have been largely avoided through the use of heuristics and preprocessing. This thesis is motivated by the challenge of including discrete reasoning directly into the estimation process.

We approach the problem by using Conditional Linear Gaussian Networks (CLGNs) as a generalization of least-squares estimation which allows the inclusion of discrete model selection variables. CLGNs are a powerful framework for modeling sparse multi-modal inference problems, but are difficult to solve efficiently. We propose the Iterative Local Model Selection (ILMS) algorithm as a general approximation strategy specifically geared towards the large scale problems encountered in tracking and mapping.

Chapter 4 introduces the ILMS algorithm and compares its performance to traditional approximate inference techniques for Switching Linear Dynamical Systems (SLDSs). These evaluations validate the characteristics of the algorithm which make it particularly attractive for applications in robot perception. Chief among these is reliability of convergence, consistent performance, and a reasonable trade off between accuracy and efficiency.

In Chapter 5, we show how the data association problem in multi-target tracking can be formulated as an SLDS and effectively solved using ILMS. The SLDS formulation allows the addition of additional discrete variables which model outliers and clutter in the scene. Evaluations on standard pedestrian tracking sequences demonstrates performance competitive with the state of the art.

Chapter 6 applies the ILMS algorithm to robust pose graph estimation. A non-linear CLGN is constructed by introducing outlier indicator variables for all loop closures. The standard Gauss-Newton optimization algorithm is modified to use ILMS as an inference algorithm in between linearizations. Experiments demonstrate a large improvement over state-of-the-art robust techniques.

The ILMS strategy presented in this thesis is simple and general, but still works surprisingly well. We argue that these properties are encouraging for wider applicability to problems in robot perception.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisors Ian Reid and David Murray for their guidance and support in the academic world.

The Active Vision Lab has provided encouragement to push through the hard times and friends to enjoy the good times. In particular, this thesis would not have been possible without hours of conversation (and coffee) with Eric Sommerlade and Gabe Sibley.

The majority of the funding for my research was provided by the Engineering and Physical Sciences Research Council with additional travel support provided by the IEEE Robotics and Automation Society and St. Anne's College.

The journey leading up to this dissertation would not have been possible without my parents and grandparents. There are no words which can describe my appreciation.

To Irina, Thank You. For your patience, for your support, for your untiring spirit.

Notation

The following general mathematical notation will be used throughout:

- $||x||_\Sigma = \sqrt{x^\top \Sigma^{-1} x}$ will denote the Mahalanobis distance/norm
- $\mathcal{N}(\mu, \Sigma)$ - multivariate normal distribution with mean μ and covariance Σ.
- $\mathcal{N}(x\,;\,\mu, \Sigma)$ - multivariate normal density at point x with mean μ and covariance Σ.
- $\mathbf{P}(A \mid B)$ - probability of A given B
- $\mathbf{E}[A \mid B]$ - expectation of A given B
- $\mathbf{Cov}[A \mid B] = \mathbf{E}\big[(A - \mathbf{E}[A \mid B])(A - \mathbf{E}[A \mid B])^\top \mid B\big]$ - covariance operator
- $\mathrm{D}_{\mathrm{KL}}(\mathbf{P}(\cdot)\|\mathbf{Q}(\cdot)) = \int_x \mathbf{P}(x) \log\Big(\frac{\mathbf{P}(x)}{\mathbf{Q}(x)}\Big)\,\mathrm{d}x$ represents the KullbackLeibler divergence.
- $\delta_y(\cdot)$ is the Dirac delta and when used in a probability density function denotes a discrete value/atom; $\int_{x\in\Omega} \delta_y(x)\,\mathrm{d}x = \begin{cases} 1 & y \in \Omega \\ 0 & \text{otherwise} \end{cases}$
- $\mathbf{P}(x \mid z) \propto f(x,z) \Leftrightarrow \mathbf{P}(x \mid z) = \frac{f(x,z)}{\sum_x f(x,z)}$
- $\tilde{f}_{\hat{x}} = f(\hat{x}) + \nabla f(\hat{x})(x - \hat{x})$ used to denote linearization of f at $\hat{x}$.
- $x \perp\!\!\!\perp y$ will be used to denote statistical independence. i.e. $\mathbf{P}(x, y) = \mathbf{P}(x)\,\mathbf{P}(y)$.
- $\mathbf{1}(x) = \mathcal{N}(x\,;\,0, \infty)$ will be used to denote the fully uninformative 'distribution' or the uniform distribution over the domain if the domain is finite. For Gaussian message passing this is a degenerate normal distribution parameterized by a 0 information matrix and vector.

Abbreviations

ILMS	Iterative Local Model Selection
SLDS	Switching Linear Dynamical System
CLG	Conditional Linear Gaussian
CLGN	Conditional Linear Gaussian Network
BP	Belief Propagation
KF	Kalman Filter
EKF	Extended Kalman Filter
IEKF	Iterated Extended Kalman Filter
KS	Kalman Smoother
IKS	Iterated Kalman Smoother
SWF	Sliding Window Filter
SLAM	Simultaneous Localization and Mapping
PGM	Probabilistic Graphical Model
PDAF	Probabilistic Data Association Filter
JPDAF	Joint Probabilistic Data Association Filter
JCBB	Joint Compatibility Branch and Bound
MCMC	Markov Chain Monte Carlo
MHT	Multi-Hypothesis Tracker
DP	Dynamic Programming
ILP	Integer Linear Program
LP	Linear Program
MRF	Markov Random Field
GRF	Gaussian Random Field
MAP	Maximum a Posteriori
HMM	Hidden Markov Model
LDS	Linear Dynamical System

Publications from this thesis

- **Latent Data Association: Bayesian Model Selection for Multi-target Tracking**
  A V Segal and I D Reid
  Proc Int Conf on Computer Vision Dec 3-6 2013, Sydney, Australia
  — Presented in Chapter 5

- **Hybrid Inference Optimization for Robust Pose Graph Estimation**
  A V Segal and I D Reid
  Proc IEEE/RSJ Int Conf on Intelligent Robots and Systems Sep 14-18, Chicago IL, USA
  — Presented in Chapter 6

1

Introduction

1.1 Motivation

Perception of the world is one of the key research topics in AI, Computer Vision, and Robotics. At it's core, the goal is to build an internal representation of the external world using the available sensors. In AI and Robotics, interpretation of the world is a major part of the perceive-plan-act loop of an intelligent system. In Computer Vision, perception of the world through cameras is the definition of the field.

A large body of research has focused on the problem of mapping from a moving platform. In the robots community this has taken the form of *Simultaneous Localization and Mapping* (SLAM) – the task of building maps from a moving robot. The problem can be summarized by the following question: given data about the world collected from a moving platform, how can we solve for the structure (mapping) of the world together with the motion of the platform (localization)? Recent research has been successful in solving this problem using a variety of methods optimized for different world models, sensors, and situations. State of the art SLAM systems can now work in real-time with both laser and visual sensors, localize the vehicle they're mounted on, and map novel parts of the environment accurately and even robustly. There are still many open problems, but the successes in handling static environments have opened new areas of research going beyond the original

problem formulation.

A current state-of-the-art robot is capable of mapping and navigating a mostly static and constrained environment. Autonomous vehicles are now almost capable of operating on a public roads, taking care to follow the rules, and avoiding collisions. These systems, however, still fall short of anything which could be called intelligent decision making. To be truly autonomous such robotic systems must eventually see the world as more than just a collection of points and landmarks. This suggests a long term research goal of richer mapping and tracking algorithms which incorporate semantic understanding.

In the short term, the field of robot perception is filled with practical difficulties which require some form of discrete reasoning about the world. The classic examples are data association and outlier rejection, two problems which have plagued researchers for decades. In this case the discrete reasoning is often a nuisance to be circumvented in order to estimate the robot pose and map. In other situations, the discrete aspects are the primary objects of interest and cannot be avoided. Object detection and tracking is one example where the discrete label is in fact one of the things we would like to know. More expressive discrete models could, for instance, use knowledge of the typical motions of people, cars, and bicycles to predict where they are likely to be in the future. These examples suggest the need for a perception system which can choose among various models to better explain the world.

The research presented in this dissertation is motivated by these higher level aspects of the mapping problem. The goal is to incorporate discrete decision making as a first class component of the perception pipeline rather than on the periphery. We accomplish this by incorporating discrete variables into the traditionally continuous estimation problems used for perception. For the remainder of the thesis, we will refer to this strategy as a hybrid discrete-continuous inference approach or equivalently as model selection.

1.2 Approach

We propose the use of Switching Linear Dynamical Systems (SLDS) and Conditional Linear Gaussian Networks (CLGN) as a modeling framework for perception. The SLDS is a fusion of a discrete Hidden Markov Model (HMM) and a continuous Linear Dynamical System (LDS). The discrete HMM serves as a model selection prior over the LDS and allows switching between multiple continuous models. CLGNs are a generalization of the same principle to a broader class of graphical models. The CLGN is the fusion of a discrete and continuous graphical model in which the discrete layer again switches between various continuous models.

The SLDS and CLGN are not novel representations, but have received limited practical attention partly due to the difficulties of inference in these models. In this thesis we introduce the *Iterative Local Model Selection* (ILMS) algorithm as an approximate inference strategy for solving the hybrid inference problems encoded by these models. ILMS is a modification of the *Belief Propagation* (BP) algorithm which selects the best local model as part of the message passing procedure. Although the model selection is greedy, the choice is iteratively revisited as the algorithm proceeds. The approach is broadly similar to *Expectation Propagation* (EP) where a local approximation is iteratively refined. Unlike Expectation Propagation, however, ILMS has guaranteed convergence and in practice tends to do so quickly.

We believe the algorithm offers an attractive compromise between speed and accuracy which is well suited for problems in robot perception. The local nature of the algorithm makes it a good choice for problems where discrete decision making is local, but the continuous variables are globally correlated. The applications addressed in this thesis, multi-target tracking and robust estimation are good examples of such problems. With the *Latent Data Association* parameterization of multi-target tracking discussed in Chapter 5, data association can be treated as a local discrete estimation problem. The continuous states of each tracked target across the entire time series are strongly correlated, but the discrete

data association variables are only related to each other through the chain of continuous variables representing each target's state.

Similarly, the robust pose graph estimation application in Chapter 6 is a good use case. The continuous variables in a pose graph are strongly correlated with each other via the measurements. The discrete inlier/outlier decisions about each individual loop closure, however, are not as strongly correlated because outlier loop closures can largely be determined based on their incompatibility with the odometry data. The compatibility of the loop closures among themselves is also important, but can be treated as a secondary concern. In other words, the 'known' data associations provide a good enough estimate of the system state so that additional data can be incorporated piecemeal without explicit consideration of all possible combinations of inliers and outliers. Many problems in Computer Vision and Robotics which use heuristics to make decisions about *how* to incorporate data fall into this category. These are good candidates for ILMS since the algorithm allows data to be incorporated locally, but still provides some measure of smoothing and re-interpretation of previously processed data in light of new evidence.

The proposed approach may not work as well for applications which require too much coordination between different sets of discrete variables. In such cases the algorithm may still be applicable if care is taken to structure the problem such that strongly correlated discrete variables appear in the same local optimization sub-problem. Estimation problems where distant discrete variables are strongly correlated among themselves, however, are fundamentally not well suited for this strategy. Structuring these problems to bring the strongly correlated variables together will remove the advantages of sparsity and make them intractable.

It is also possible for the algorithm to fail due to its local nature. In the application to robust pose graph estimation, for example, each loop closure is considered individually based on its compatibility with the current pose graph estimate. It is possible to imagine an outlier loop closure which is 'almost correct' being interpreted as an inlier and precluding

the correct (but now incompatible) loop closure from being used. This could result in a string of off-by-one loop closures being selected due to self-consistent perceptual aliasing.

Despite these limitations, ILMS has significant advantages over standard estimation techniques — it can handle much richer models in a principled manner while scaling to realistically sized environments. In the remainder of this section we will describe how CLGNs can be seen as a generalization of least-squares estimation and outline the ILMS algorithm.

1.2.1 Continuous State Estimation and Gaussian Random Fields

The typical formulation of state estimation for mapping involves defining variables for the geometric properties of the map and objectives which define the relationship between them. The set of objectives and variables define a sparse non-linear least squares problem which is solved iteratively using the Gauss-Newton algorithm or an equivalent numerical optimization scheme. In the case of Gauss-Newton, each linearization results in a sparse quadratic optimization problem which must be solved to compute the next linearization point. Since the optimization problem may involve tens of thousands of variables, it is important to take advantage of sparsity. This is typically handled by using one of several strategies from sparse linear algebra. Alternatively, the quadratic optimization problem can be interpreted as a Gaussian Random Field (GRF). In this case Belief Propagation can be used as a sparse inference algorithm for solving the least squares problem. Once we have switched to the inference perspective, it becomes natural to incorporate model selection by adding discrete variables to the problem. Generalizing the GRF into a Conditional Linear Gaussian Network is one way to accomplish this.

1.2.2 Switching Models for Tracking and Mapping

Having added the possibility of discrete modeling into our estimation algorithm, we now consider what this actually gets us. In this section we will sketch out several possible

applications of switching models and attempt to demonstrate that they are more powerful than they may initially appear.

As a simple use case, we can make our estimation robust by using a Mixture-of-Gaussians observation model with a high variance component for outliers. This corresponds to adding an inlier/outlier indicator variable for each observation. To model streaks of outliers, such as those caused by occlusions, the indicator variables can be coupled together with a prior which captures this tendency.

In certain situations, data association can also be treated as a switching model by using a single 'compound' discrete variable with a combinatorial number of states (one for each valid association). Although this may seem impractical, we will show that approximate inference on such a model is sometimes possible without enumerating all discrete states.

We can also consider behavior modeling of dynamic objects – a more interesting situation where the discrete variables are a priori correlated. A pedestrian may have a variety of different behaviors depending on the situation. These would correspond to a set of motion models describing the motion in each case. Since the behavior at each time is not independent, there are correlations between the time-adjacent model selection variables within each track.

Many other situations could potentially be modeled using switching networks. An autonomous vehicle may have multiple motion models corresponding to different road surfaces. An unexpected sensor reading may be interpreted as either a failure of the sensor itself or a drastic change in the state of the world. Each of these possibilities corresponds to a switching model. This perspective turns the problem of discrete reasoning into one of inference in switching models.

1.2.3 Iterative Local Model Selection

The main contribution of this thesis is the Iterative Local Model Selection (ILMS) algorithm for approximate inference in CLGNs. Iterative Local Model Selection is an approximate

Belief Propagation algorithm which relies on the observation that many practical model selection problems are fundamentally local. That is, the discrete variables involved are primarily correlated with a small neighborhood of other variables.

To explain the algorithm at an intuitive level, consider why the exact Belief Propagation algorithm becomes intractable for hybrid inference problem. The problem occurs when the computation of a message requires marginalizing over a discrete variable, resulting in a Mixture-of-Gaussian message. ILMS avoids this situation by locally picking the 'best' value for the discrete variable and computing the outgoing message conditioned on this value. Unlike the standard Belief Propagation algorithm, ILMS no longer converges after a single upward and downward pass through the clique tree and so requires multiple rounds of message passing to achieve convergence. The algorithm is 'local' because only a local neighborhood of discrete variables is considered during each message computation. We refer to the algorithm as 'iterative' because the local model selection is revisited multiple times until the algorithm converges to a consistent solution.

ILMS performs a local search through the discrete space and can be thought of as a very specific variant of block coordinate ascent where each block contains a small set of discrete variables together with all continuous variables. The intuition can be illustrated by a hypothetical linear least-squares problem with outliers. In this problem we want to estimate the mean of a single variable with two possible models for each observation. A single binary variable is associated with each observation to indicate the outliers. For a particular indicator d_i, we hold the others fixed and pick the best value for d_i by solving the least squares problems correspond to the two possible values $d_i = 0$ and $d_i = 1$. ILMS cycles through all of the indicator variables updating them in this manner one by one. Cycling in a particular order and using message passing allows the algorithm to avoid recomputing the least-squares solution multiple times by updating previous solutions instead of starting from scratch.

1.3 Contributions

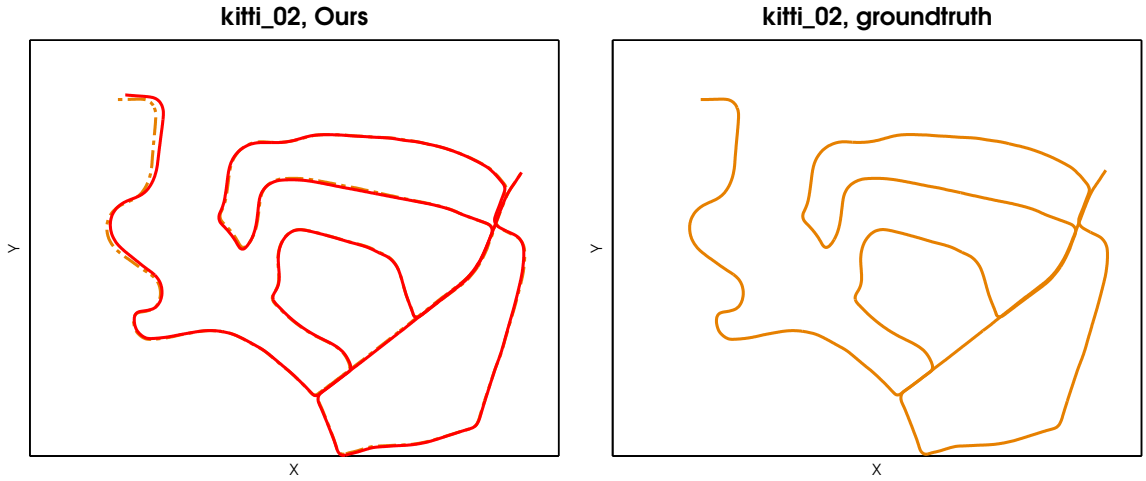
The novel contributions of this thesis consist of the ILMS inference algorithm and two practical applications. The algorithm itself is presented in in Chapter 4 in the context Switching Linear Dynamical Systems. We formally prove the convergence of the algorithm and demonstrate its advantages compared to existing approximate inference methods. This initial evaluation is conducted on a set of synthetic tracking problems where exact ground truth is known.

In Chapter 5, we show how ILMS can be used for multi-target tracking. The resulting *Latent Data Association algorithm* is an approach to multi-target tracking which recasts data association as an SLDS. For this application, ILMS is used to infer both data associations and some discrete properties of the individual targets. We include target-specific discrete states to model track termination, outlier trajectories, and missing observations. Evaluations are performed on real-world data where Latent Data Association is shown to match state-of-the-art performance as of publication in 2013.

Chapter 6 introduces an application of ILMS for Robust Pose Graph SLAM. We explicitly introduce discrete variables to model outlier loop closure constraints in the graph, so that ILMS can infer when an appearance based loop closure detector may have failed. Because pose graphs are not linear chains of variables, we use a Conditional Linear Gaussian Network model to represent the resulting hybrid inference problem. This requires a generalization of the ILMS algorithm to clique tree structured models which is possible due to a novel message passing order. Because pose graph estimation is a non-linear estimation problem, we present a variant of the Gauss-Newton algorithm which uses ILMS instead of a least-squares solver in the inner loop. The resulting *Hybrid Inference Optimization* algorithm demonstrates state-of-the-art performance on publicly available datasets as of publication in 2014. We note that we were able to either match or beat the state-of-the-art in both of the presented applications despite using a much more general approach than used by competing problem-specific algorithms. Fig. 1.1 highlights results from the two

applications.

(a) Chapter 5: Multi-Target Tracking.



(b) Chapter 6: Robust Pose Graph Estimation.

Figure 1.1: Illustration of results on tracking and mapping applications.

1.4 Overview

The thesis is divided into seven chapters. The following two chapters will cover previous work in the field and introduce relevant background material as well as the notation which will be used in the remainder of the thesis. Following these, we will present novel research in chapters 4-6. The final chapter will draw conclusions and present topics of interest for future work. The following list summarizes each chapter:

- Chapter 1 consists of this introduction.
- Chapter 2 outlines related work
- Chapter 3 reviews background material on Non-linear Optimization, Sparse Estimation, Probabilistic Graphical Models, and Gaussian Belief Propagation.
- Chapter 4 introduces the Iterative Local Model Selection (ILMS) algorithm from a theoretical perspective. This chapter also validates the algorithm on several synthetic problems.
- Chapter 5 applies the ILMS algorithm to the problem of multi-target tracking from a stationary platform.
- Chapter 6 applies a generalization of the algorithm to robust pose graph SLAM.
- Chapter 7 draws conclusions from the research conducted and outlines possible avenues for future work.

2

Related Work

Before introducing novel research, we will start by reviewing the overall progression of the field. The more specific recent work will be presented in the chapters where it is most relevant. Two areas of previous work are directly related to this thesis. The first, *Multi-Target Tracking*, focuses on tracking moving objects from a static reference frame. The second, SLAM, deals with tracking a moving sensor platform through an unknown environment. Although today these are different communities, both fields historically trace their origins to seminal work on state estimation from the 1960s. In both cases, discrete variables and various forms of model selection have play an important and challenging role. In multi-target tracking, the fundamental difficulty is associating observations with targets and estimating the number of targets in the environment. In the classic SLAM formulation, data association is often simpler since we are not concerned with a changing environment, but is nonetheless challenging. We take a unified view of SLAM and multi-target tracking as sparse state estimation problems with discrete model selection components. Given this perspective, it is natural to review the background in both of these areas as well as their connections, before considering a framework for model selection. To this end, we begin by reviewing the early literature on state estimation, followed by multi-target tracking and the history of the SLAM problem in Robotics.

2.1 State Estimation and Dynamical Systems

Modern state estimation in dynamical systems was pioneered in the late 1950s and early 1960s for tracking missiles and orbiting satellites. The original problem formulation consisted of a target being tracked and a series of noisy observations of that target. Given a motion model of the target and a sensor model for the observations, the goal is to estimate the true motion. The state of the system at time t is represented by the state vector x_t. Observations are denoted z_t and a continuous *Hidden Markov Model* (HMM) is used as the system model.

The *Kalman Filter* (KF) [5] was the first practical and real-time estimation algorithm capable of effectively tracking these targets. The key insight of the KF was that all information about the past history of observations can be captured by the current state estimate combined with its uncertainty. While Kalman's original work was presented using different notation, with modern understanding the Kalman filter can be seen as the simplest case of a Bayes filter[6, 7]. The KF at time t consists of a Gaussian distribution over x_t conditioned on all previous observations.

$$\mathbf{P}(x_t \mid z_{1:t}) = \mathcal{N}\big(x_t\,;\, \hat{\mu}_{x_t}, \hat{\Sigma}_{x_t}\big) \tag{2.1}$$

At time $t+1$, the filter can be updated to include the measurement z_{t+1} using update equations based on Bayes' rule

$$\mathbf{P}(x_{t+1} \mid z_{1:t}) = \int \mathbf{P}(x_{t+1} \mid x_t)\, \mathbf{P}(x_t \mid z_{1:t})\, \mathrm{d}x_t \tag{2.2}$$

$$\mathbf{P}(x_{t+1} \mid z_{1:t+1}) = \frac{\mathbf{P}(x_{t+1} \mid z_{1:t})\, \mathbf{P}(z_{t+1} \mid x_{t+1})}{\int \mathbf{P}(x_{t+1} \mid z_{1:t})\, \mathbf{P}(z_{t+1} \mid x_{t+1})\, \mathrm{d}x_{t+1}} \tag{2.3}$$

If both the motion and observation models are linear with Gaussian noise, these updates can be computed exactly in closed form.

Other variants of the KF algorithm were quickly discovered in the following decade. Better numerical stability was observed by using the square root covariance matrix $\Sigma = \Sigma^{\frac{1}{2}}{\Sigma^{\frac{1}{2}}}^{\top}$ [8, 9] or alternatively the square root information matrix $\Sigma^{-1} = I^{\frac{1}{2}}{I^{\frac{1}{2}}}^{\top}$ [10].

The smoothing problem can be defined as the use of observations from $t > t_0$ in order to compute an estimate of the system state at t_0 [11]. Unlike the filtering problem, where we are maintaining a running estimate of the posterior $\mathbf{P}(x_t \mid z_{1:t})$, in smoothing we are attempting to estimate $\mathbf{P}(x_t \mid z_{1:T})$ where $T > t$ is some time in the future. Early work on state estimation also discovered efficient algorithms for performing this task [12, 13]. The *Kalman Smoother* (KS) algorithm is based on running two separate filters, one forward and one backward in time. At any given time, the forward and backward filter estimates can be combined to generate the optimal maximal likelihood estimate conditioned on both past and future observations. This algorithm and its generalizations are closely related to sparse non-linear estimation problems and inference in graphical models.

Kalman's initial work on linear systems was quickly followed by extensions to non-linear systems via the *Extended Kalman Filter* (EKF) [14, 15]. In this case the motion and observation models contain a non-linear mean function plus Gaussian noise.

$$\mathbf{P}(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1}\,;\, f(x_t), \Sigma_{\text{mot}}) \tag{2.4}$$

$$\mathbf{P}(z_t \mid x_t) = \mathcal{N}(z_t\,;\, h(x_t), \Sigma_{\text{obs}}) \tag{2.5}$$

The EKF is a simple extension of the Kalman filter where the non-linear functions f and h are linearized at the estimated system state prior to performing the updates in Eq. 2.2 and Eq. 2.3. In the following set of EKF equations, we use $\hat{x}_t$ to denote the state estimate, and $\tilde{\mathbf{P}}, \tilde{\mathbf{E}}$ to denote the approximate linearized distributions and expectations respectively. The linearized functions are represented by $\tilde{f}$ and $\tilde{h}$. Given the previous state estimate $\hat{x}_t$,

$$\tilde{f}_{\hat{x}_t}(x_t) = f(\hat{x}_t) + \nabla f(\hat{x}_t)(x_t - \hat{x}_t) \tag{2.6}$$

$$\tilde{\mathbf{P}}(x_{t+1} \mid x_t) = \mathcal{N}\big(x_{t+1}\,;\, \tilde{f}_{\hat{x}_t}(x_t), \Sigma_{\text{mot}}\big) \tag{2.7}$$

$$\hat{x}_{t+1} = \tilde{\mathbf{E}}[x_{t+1} \mid z_{1:t}] \tag{2.8}$$

$$\tilde{h}_{\hat{x}_{t+1}}(x_{t+1}) = h(\hat{x}_{t+1}) + \nabla h(\hat{x}_{t+1})(x_{t+1} - \hat{x}_{t+1}) \tag{2.9}$$

$$\tilde{\mathbf{P}}(z_{t+1} \mid x_{t+1}) = \mathcal{N}\big(z_{t+1}\,;\, \tilde{h}_{\hat{x}_{t+1}}(x_{t+1}), \Sigma_{\text{obs}}\big) \tag{2.10}$$

$$\hat{x}_{t+1} = \tilde{\mathbf{E}}[x_{t+1} \mid z_{1:t+1}] \tag{2.11}$$

Improved accuracy can be achieved by iteratively re-linearizing; this version is known as the *Iterated Extended Kalman Filter* (IEKF) and has been applied since at least the 1970s [11]. Bell and Cathey [16] demonstrate that the IEKF is equivalent to linearizing the system after using the Gauss-Newton algorithm to obtain a local MAP estimate of x_{t+1}

$$\tilde{\mathbf{P}}(x_{t+1} \mid x_t) = \mathcal{N}\big(x_{t+1}\,;\, \tilde{f}_{\hat{x}_t}(x_t), \Sigma_{\text{mot}}\big) \tag{2.12}$$

$$\hat{x}_{t+1} = \underset{x_{t+1}}{\operatorname{argmax}} \left\{ \mathbf{P}(z_{t+1} \mid x_{t+1})\, \tilde{\mathbf{P}}(x_{t+1} \mid x_t)\, \tilde{\mathbf{P}}(x_t \mid z_{1:t}) \right\} \tag{2.13}$$

$$\tilde{\mathbf{P}}(z_{t+1} \mid x_{t+1}) = \mathcal{N}\big(z_{t+1}\,;\, \tilde{h}_{\hat{x}_{t+1}}(x_{t+1}), \Sigma_{\text{obs}}\big) \tag{2.14}$$

For the *Iterated Kalman Smoother* (EKS), a similar equivalence can be made to Gauss-Newton applied to the full system state vector[17]

$$\hat{x}_{1:t} = \underset{x_{1:t}}{\operatorname{argmax}} \left\{ \mathbf{P}(z_1 \mid x_1)\, \mathbf{P}(x_1) \prod_{t=2,\ldots,T} \mathbf{P}(z_t \mid x_t)\, \mathbf{P}(x_t \mid x_{t-1}) \right\} \tag{2.15}$$

Bell [17] demonstrate that the EKS can be thought of as a sparse decomposition of the global least squares problem via the Matrix Inversion Lemma. Bertsekas [18] show an equivalence to incremental least squares and provide a proof of convergence subject to some constraints on the step size.

More recently the *Unscented Transform* [19, 20] has been proposed as an alternative to the linearization in the EKF. The resulting algorithm, referred to as the *Unscented Kalman Filter* (UTF), is based on representing the distribution $P(x_t|z_{1:t})$ as a deterministic set of weighted sample points

$$\mathbf{P}_{\text{unscented}}(x_t \mid z_{1:t}) = \sum_i w_{ti} \cdot \delta_{\mathcal{X}_{ti}}(x_t) \tag{2.16}$$

such that the sample mean and covariance match the target distribution. Instead of linearizing the motion and observations models, the functions f_t and h_t are applied directly to the sample points to generate a transformed set of points

$$\mathcal{Y}_{ti} = f(\mathcal{X}_{ti}) \tag{2.17}$$

The effect of f on the distribution can be estimated by computing the weighted mean and covariance of the transformed points. Iterated versions of the UKF have also been explored [21].

The need to deal with discrete variables and model selection was also recognized early in the field. In order to deal with an unknown system model, Magill [22] introduced what has become known as the *Multiple Model* (MM) approach[11]. MM runs multiple independent KFs in parallel using the same input data, each with a separate system model. At each point in time, the output of the MM filter is a linear combination of the filtered estimates from the bank of filters. The process is effectively being modeled as a mixture of several independent linear Gaussian models. Crucially, the MM filter does not model a switching *process*; it assumes there is a single linear model which explains the data, but we are are not sure which model it is.

Ackerson and Fu [23] appear to be the first to propose a switching filter which explicitly models transitions between different discrete states. They propose a KF operating in a set of different noise environments. A discrete Markov chain models which environment the system is in at any given time. Each environment corresponds to different noise parameters in the motion and observation models.

$$k_t \in \{1, \ldots, K\} \tag{2.18}$$

$$\mathbf{P}(k_{t+1} = i \,|\, k_t = j) = p_{ij} \tag{2.19}$$

$$\mathbf{P}(x_{t+1} \,|\, x_t, k_{t+1} = i) = \mathcal{N}\Big(x_{t+1}\,;\, f_t(x_t), \Sigma^{(i)}_{\text{mot}}\Big) \tag{2.20}$$

$$\mathbf{P}(z_t \,|\, x_t, k_t = i) = \mathcal{N}\Big(z_t\,;\, h_t(x_t), \Sigma^{(i)}_{\text{obs}}\Big) \tag{2.21}$$

They observe that optimal filtering is not feasible in such a system because the true representation for the posterior $P(x_t|z_{1:t})$ is a mixture with K^t components. An approximation is proposed where the filtering state is represented by a single Gaussian distribution. Tugnait [24] present a survey of approximate algorithms for such *Switching Linear Dynamical Systems* (SLDS) where the process parameters change according to a discrete Markov process. They note several approximation schemes similar to Ackerson and Fu [23] which had been proposed up to that point in time.

Blom [25] introduced the *Interacting Multiple Model* (IMM) filtering approach as an extension of the MM filter. This approach is similar in spirit to Ackerson and Fu [23]. Instead

of approximating $\mathbf{P}(x_t \mid z_{1:t})$ as a single Gaussian, however, the posterior is approximated as a mixture of Gaussians over the single discrete variable k_t

$$\sum_i \tilde{\mathbf{P}}(x_t \mid k_t = i, z_{1:t})\, \tilde{\mathbf{P}}(k_t = i \mid z_{1:t}) \approx \sum_{k_{1:t}} \mathbf{P}(x_t, k_{1:t} \mid z_{1:t})\, \mathbf{P}(k_{1:t} \mid z_{1:t}) \tag{2.22}$$

2.2 Tracking

Tracking is a special case of state estimation, but with the additional difficulty of data association, cluttered 'false-alarm' sensor readings, and multiple targets. Many of these difficulties are identical to those arising in the SLAM literature. Traditional approaches to multi-target tracking were pioneered assuming point-like targets such as radar returns. Most of these approaches are progressive variations and generalizations of single target tracking in a cluttered environment.

In a tracking problem, there are typically multiple measurements recorded at each t, denoted by z_{ti} for $i = 1, \ldots, M_t$, which must be associated with the correct targets. Even if there is only one target, outlier measurements must still be filtered out and ignored.

In the simplest case *gating* can be used to determine if a candidate measurement originated from the target being tracked [26]. Given the expected observation $\hat{z}_{t+1}$ and its covariance $\Sigma_{z_{t+1}}$,

$$\hat{z}_{t+1} = \mathbf{E}[z_{t+1} \mid z_{1:t}] = \int z \cdot \mathbf{P}(z_{t+1} \mid x_{t+1})\, \mathbf{P}(x_{t+1} \mid x_t)\, \mathbf{P}(x_t \mid z_{1:t})\, \mathrm{d}z \tag{2.23}$$

$$\Sigma_{z_{t+1}} = \mathbf{Cov}[z_{t+1} \mid z_{1:t}] \tag{2.24}$$

each measurement in the new set of observations $\{z_{t+1,i}\}$ is considered valid if it passes a chi-squared test

$$(z_{t+1,i} - \hat{z}_{t+1})^\top \Sigma_{z_{t+1}}^{-1} (z_{t+1} - \hat{z}_{t+1,i}) \leq \tau \tag{2.25}$$

where τ is a chosen based on the desired confidence level and the χ^2 distribution. The valid measurement which is closest to the predicted observation is incorporated into the filter.

The *Probabilistic Data Association Filter* (PDAF), first proposed by Bar-Shalom and Tse [27], is a more sophisticated approach which allows weighted influence from all measurements. Define $\Theta_t \in \{0, \ldots, n\}$ to be the data association variable with $\Theta_t = i$ indicating z_{ti} is the valid reading at time t. If $\Theta_t = 0$, all of the measurements are invalid. Outlier measurements are assumed to be generated by a uniform distribution over the space of possible sensor readings. For each measurement a weight w_i is computed based on the likelihood of the measurement being the correct association (assuming $\tilde{\mathbf{P}}(x_t \mid z_{1:t-1})$ is an approximation of the posterior over past data associations $\Theta_{1:t-1}$)

$$w_i = \mathbf{P}(\Theta_t = i \mid z_{1:t}) = \frac{\int \mathbf{P}(z_{ti} \mid \Theta_t = i, x_t)\, \tilde{\mathbf{P}}(x_t \mid z_{1:t-1})\, \mathrm{d}x_t}{\sum_j \int \mathbf{P}(z_{ti} \mid \Theta_t = j, x_t)\, \tilde{\mathbf{P}}(x_t \mid z_{1:t-1})\, \mathrm{d}x_t} \tag{2.26}$$

Measurements are incorporated into the filter by setting $\tilde{\mathbf{P}}(x_t \mid z_{1:t})$ to a Guassian distribution with the same mean and covariance as the posterior mixture

$$\sum_{i>0} w_i\, \mathbf{P}(z_{ti} \mid \Theta_t = i, x_t)\, \mathbf{P}(x_{t-1} \mid x_t)\, \tilde{\mathbf{P}}(x_{t-1} \mid z_{1:t-1}) \tag{2.27}$$

For multi-target tracking, there are multiple target states which must be tracked in addition to multiple measurements. We will refer to the state of the j^{th} target being tracked via x_{tj}. Each measurement has a data association variable, with $\Lambda_{ti} = j$ indicating measurement i assigned to target j, and $\Lambda_{ti} = 0$ indicating an outlier/clutter measurement. The multi-target data association problem comes with the additional constraint that two measurements cannot be assigned to one target

$$j \neq l, \Lambda_{ti} \neq 0 \Rightarrow \Lambda_{ti} \neq \Lambda_{tj} \tag{2.28}$$

The *Joint Probabilistic Data Association Filter* (JPDAF) [28, 29, 30] generalizes the PDAF to take into account multiple targets. The weight of each observation is distributed among all targets, keeping in mind the additional constraint that two targets cannot be responsible for the same measurement. The set of weights is expanded to allow a separate weight for each possible target-measurement pair. In the JPDAF, w_{ij} is defined as the weight of measurement i towards target j

$$w_{ij} = \mathbf{P}(\Lambda_{ti} = j \mid z_{1:t}) \tag{2.29}$$

Once the weights have been computed, the update equations for each target are identical to the PDAF. Note that computing the exact JPDAF weights requires enumerating all possible associations, so an approximation is typically used [31, 32].

Approaches based on the PDAF/JPDAF suffer from two problems [33]. First, since the number of targets is assumed known, ad-hoc methods must be used to estimate this quantity. Second, PDAF-based trackers have a tendency to explain the same observation with multiple identical targets, a problem known as track coalescence.

The *Multiple Hypothesis Tracker* (MHT), initially proposed in Reid [34], is an alternative which does not use weights to split up measurements between multiple targets. Instead, the MHT explicitly considers all possible data associations and maintains an optimal set of trajectories conditioned on each of the association hypotheses. This results in an expanding tree of Kalman Filters: for each hypothesis present in the previous time step, we create a set of child-hypotheses, each corresponding to a different choice of data association at the current time. In order to keep this procedure practical, hypotheses are pruned based on their overall likelihood, with the top k hypotheses propagated to the next time step. Cox and Hingorani [35] introduced a more efficient version of the MHT by using a polynomial time algorithm to directly enumerate the top candidates without generating all possible combinations of data associations. Since a target is implicitly defined as a set of data association decisions, target creation/deletion is managed automatically when the hypothesis tree is pruned. Blackman [36] provides a summary of the *MHT* and its extensions.

Streit and Luginbuhl [37] proposes a modification of the JPDAF to achieve a similar delayed decision making affect as the MHT. An EM-based weighting procedure is performed within a sliding window in order to allow past data associations to change as more information becomes available. The down side of this approach compared to the *MHT* is the need to explicitly decide when to create and delete target tracks.

In general, the data association problem can be viewed as a special case of state estimation in switching environments. In this case, the switching occurs in the combinatorial

space of all possible data associations. Just as in the case of switching dynamical systems, the possible data associations create a mixture model posterior. Many of the above data associations algorithms amount to various approximations of this mixture.

2.3 Mapping

2.3.1 SLAM

The question posed by *Simultaneous Localization and Mapping* (SLAM) is: given a series of sensor measurements, can we estimate the set of robot/sensor positions at which they were made, together with the structure of the world which causes the sensor readings. In the Robotics context, sensor measurements can take the form of odometry, GPS, lasers, or camera images recorded at a sequence of points in time. In the context of Computer Vision, the measurements are the images recorded by the camera, and possibly the output of an accelerometer. At it's core, SLAM is a generalization of classical state estimation problem to include a map $M = (m_1, \ldots, m_n)$ composed of the landmarks $\{m_i\}$. Unlike the system/robot state x_t, the map is not time dependent, but a fixed property of the world to be estimated. Further, the observations no longer depend solely on the system state x_t, but also on the unknown map M.

The data association problem faced by SLAM is similar to the one in multi-target tracking: multiple observations must be matched with multiple landmarks. A major distinction, though, is that the posterior distributions of the landmark states are correlated via the moving sensor platform. This makes it challenging to resolve data association without taking into account the global correlation structure of the posterior. Maintaining this posterior as part of an EKF approach, for example, requires maintaining the posterior covariance of (x_t, M) which is quadratic in the size of M. Since landmarks are typically added into the map, this approach quickly becomes intractable for larger environments. The need to efficiently represent the correlated posterior of the SLAM problem has lead to research into sparse representations and approximations.

The first formulation of the SLAM problem in a stochastic setting is generally credited to Smith et al. [38], whose approach was based on the EKF and a relative transformation graph first proposed in Smith and Cheeseman [39]. Smith et al. formulate the SLAM problem as the estimation of a network of transformations between coordinates frames. The estimated network of transformations was maintained using an EKF with an associated covariance matrix representing correlations between the different frames. Moutarlier and Chatila [40] developed an alternative formulation of the stochastic map and demonstrated an implementation of their system in practice. In both cases data association is side stepped by assuming a known mapping between map elements and sensor observations.

Leonard and Durrant-Whyte [41, 42] proposed an EKF formulation which avoided representing the full correlation structure of the map by separating the landmarks used for mapping and localization. A set of well-localized landmarks are used for localization while new landmarks are separately tracked until their location can be accurately estimated. The authors critiqued the stochastic map approach as impractical due to the computation complexity of tracking the full robot and landmark state space with an EKF.

Dissanayake et al. [43] provides a review of the early literature on EKF SLAM, a theoretical investigation of the problem from a state estimation perspective, and a proof of concept implementation. Dissanayake et al. [43] and Bailey et al. [44] as well as [45, 46] investigate the problem of inconsistency in the EKF SLAM algorithm. The overall conclusion is that EKF SLAM will inevitably fail if the uncertainty in vehicle position is not kept low. This failure is caused by incorrect and inconsistent linearization being incorporated into the filter.

Davison and Murray [47] was the first successful system to use vision as a sensor for real-time SLAM. Instead of processing the whole image frame, Davison and Murray directed computational effort by actively looking for each feature in the image within the region where it is expected to be found. This helped resolve two issues with EKF-SLAM. First, the active search framework allowed a smaller number of landmarks to be carefully

tracked, rather then extracting large numbers of features and initializing a large number of new landmarks at each frame. Second, since EKF-SLAM kept track of landmark position uncertainty, projecting the current uncertainty ellipse onto the image plane placed a bound on where the landmark was likely to be observed. By only searching within these bounds, efficiency was greatly increased.

[48] investigates a distributed version of SLAM and suggests the use of the Information Filter over the more standard Kalman Filter. The information parameterization allows incorporating novel information by simply adding constraints to the representation without performing any matrix inversions. This approach is particularly useful for multi-robot problems, but has advantages for standard SLAM as well.

An alternative formulation of SLAM, known as pose graph estimation, was proposed by Lu and Milios [49]. The pose graph approach removes landmarks from the map entirely and treats the problem as one of estimating a network of robot positions. Constraints between poses can be generated by using scan matching [50] to align the observed sensor readings. The network of constraints, referred to as a pose graph, is directly optimized. In contrast to EKF SLAM, where only the most recent pose is tracked, pose graph SLAM updates the full trajectory of the robot and so avoids permanently incorporating inaccurate linearizations. The Sparse Extended Information Filter (SEIF) [51] combines the advantages of the Information Filter parameterization with the pose graph formulation of Lu and Milios [49]. Nuchter et al. [52] extends the Graph SLAM approach to 3D environments by using laser generated pointclouds as input and the Iterative Closest Point (ICP) [50] algorithm for scan matching.

Montemerlo et al. [53] introduced the FastSLAM algorithm which is based on a combination of sampling and the EKF. The key realization was that conditioning on the robot trajectory made all of the landmarks conditionally independent. By representing the distribution over trajectories as a finite set of samples, the locations of the landmarks became conditionally independent when conditioned on a single trajectory sample. This allowed

FastSLAM to avoid the $O(n^2)$ running cost of EKF-SLAM at the expense of sampling the space of trajectories. This approach was extremely successful for robots constrained to a plane and equipped with range sensors such as laser scanners and sonar. Sim et al. [54] adapted the FastSLAM approach to a visual setting using a stereo rig and SIFT features [55].

Davison et al. [56] introduced MonoSLAM, the first real-time monocular SLAM algorithm. Unlike previous work which relied on either accurate odometry estimates, or slow sensor movement, the algorithm introduced by Davison et al. works on a rapidly moving hand-held camera without any odometry. An active search procedure similar to Davison and Murray [47] is used to locate features, and a delayed initialization scheme is used to help estimate the position of landmarks. Since a monocular camera cannot give a 3D position estimate when a feature is first observed, Davison et al. used a collapsed particle filter to estimate a distribution of possible landmark locations along a ray. When this distribution is sufficiently close to being Gaussian, the landmark is fully initialized and tracked with EKF-SLAM.

Eade and Drummond [57] present a monocular FastSLAM as an alternative to EKF-SLAM. Eade and Drummond combined the top-down active search of [47], with an improved version of the delayed initialized procedure in [56]. The improved initialization procedure uses an inverse-depth parameterization to sample particles along an initial ray. Unlike MonoSLAM, the particles are used to estimate camera position even before the landmark is fully initialized.

2.3.2 Structure from Motion

Structure-from-Motion is a similar, but historically independent estimation technique used for offline map-building. Unlike research in SLAM, which has focused on providing a real-time solution to the online version of the problem, SfM has focused on offline map estimation from a large set of images. This procedure traces its history back to the early

19th century roots of *Photogrammetry* [58], which uses photographs to estimate topography for map-making. The Photogrammetry community had coined the term *bundle adjustment* (in reference to adjusting the bundles of rays going from features to camera planes) to refer to the process of least squares estimation of these maps based on matching features between photographs.

A modern understanding of the procedure from the computer vision perspective is presented in the survey by Triggs et al. [59], as well as Hartley and Zisserman [60]. We will define $\{X_j\}$ as a set of 3D points whose position is unknown. These points will loosely correspond to the M variable in the SLAM formulation. The points are observed through a set of cameras, which will be known via their set of projection operators $\{P^i\}$. The application of a camera projection operator, P^i, onto a 3D point, X_j, yields a 2D measurement of that point, $x_j^i = P^i(X_j)$.

Once we are given the set of observations $\{x_j^i\}$, we can define the bundle adjustment problem as least-squares estimation of all of the camera parameters together with the locations of the 3D points (landmarks in the SLAM terminology):

$$\underset{\{P^i\},\{X_j\}}{\operatorname{argmin}}\left(\sum_{i,j} ||P^i(X_j) - x_j^i||_\Sigma^2\right) \tag{2.30}$$

where $||\cdot||_\Sigma$ is the L2 norm using Σ as the metric.

Agarwal et al. [1] demonstrate a very impressive and large scale implementation of state of the art structure from motion. They build a large scale reconstruction of the city of Rome using over 1,000,000 individual uncalibrated photographs obtained from user content uploaded to Flickr. Fig. 2.1 shows a sample reconstruction from the group's website.

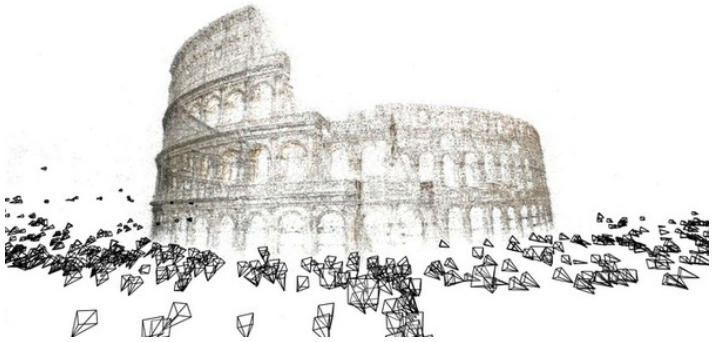


Figure 2.1: A reconstruction of the Colosseum in Rome obtained from Agarwal et al. [1]

2.4 Sparsity and Decomposition

The topic of sparsity and decomposition deals with efficient computational methods which take advantage of the sparse relationships between variables in problems. Such methods are required for both mapping and tracking because the full state spaces of these problems often involves tens of thousands of variables. For tracking problems, the Kalman Filter and Smoother can be thought of as sparse estimation algorithms. In multi-target tracking, the fact the each track is represented as an independent Kalman filter is also a form of sparsity.

In the SfM and SLAM communities, a large body of research has targeted efficient decompositions of the estimation problem which are applicable to large data sets. The idea behind this line of research is to partition a large mapping problem into multiple smaller problems which can be more easily handled. The local maps can then be merged into a global map estimate. There are two practical advantages to this approach. First, it allows maps larger than the memory of the computer to be built. Second, the algorithm is more efficient since it ignores the generally weaker dependencies between distance map features. Maintaining the full cross covariance between distance landmarks is the main cause of the quadratic memory footprint of EKF-SLAM. Much like the Sparse Extended Information Filter (SEIF) [51], we can gain a computational advantage by maintaining these cross correlations only when they are strong. Whereas the SEIF does this by examining individual entries in the dependency graph for relevance, submapping approaches to SLAM and SfM take advantage of the natural tendency of distance landmarks to be weekly correlated.

Chong and Kleeman [61] introduced one of the first submapping schemes where local maps were built up for each region of the environment. Each local map was treated as an independent mapping problem with a reference frame based on the initial robot position at the 'start' of the map. Within each local map, a full covariance estimate of all landmarks is tracked using EKF SLAM. Landmarks which are not part of the same submap do not have an explicit representation of their relative covariance stored. Instead, the covariance between the local reference frames themselves is maintained. Leonard and Feder [62] use a similar scheme, except with a global reference coordinate the offsets between submaps. This simplifies finding the appropriate submap for the current robot pose since the locations of the submaps themselves are stored in a single reference frame.

In Bosse et al. [63] each local submap is stored in its own coordinate system, with a network of relative transformations between the local frames. This is similar in principle to the original stochastic map, but here each vertex represents an entire submap rather than a single map feature. The network of coordinate frames is undirected with loops allowed. Dijkstra's algorithm is used on this network to find chains of transformation between different coordinate frames. The algorithm maintains a set of hypothesis local maps and tracks the robot in parallel in each of the hypothesis. This avoids relying on a single local map and allows smooth transitions to different local representations by changing the active set of hypotheses. Periods of simultaneous tracking within multiple hypotheses are used to refine the estimated transformations between the corresponding submaps.

Sibley et al. [64], Mei et al. [65] introduce a system based on the natural limit of the submapping approach of Bosse et al. [63]. In this case each individual robot pose is regarded as a separate submap containing only the landmarks which were first observed at that time. This allows constant time operation since incorporating new observations into the problem does not effect any variables beyond a relatively small set of poses in the past. Specifically, adding observations of a landmark induces correlations with all other robot poses which have observed that landmark though the chain of relative transformations between these poses. Since all transformations are stored in a relative representation, any

pose not on this chain becomes independent of the new measurements.

Ni et al. [2] uses a local parameterization for each submap, but does so in a smoothing rather than filtering framework. The resulting *Tectonic SAM* algorithm can be interpreted as a non-linear optimization algorithm composed of two types of objectives. The intra-map objectives are those which constraint variables entirely contained within a single submap. Because these variables are parameterized in a local coordinate frame, these objectives do not depend on any variables outside of their submap. The second set of objectives are inter-map objectives which constraint variables spanning more than one submap. The values of the inter-map objectives are affected by the transformations relating the local coordinate frames. Ni et al. [2] propose an optimization algorithm which alternates between optimizing the local submaps (intra-map objectives) and the objectives connecting different submaps (inter-map objectives). In the first phase, the submap solutions are held fixed, while the inter-map transformations are optimized. In the second phase, the inter-map transformations are held fixed while the submaps themselves are optimized. These two optimization problems are shown to be loosely coupled because of the local parameterization within each submap. A key point of this approach is that Jacobians can be re-used between the two phases. Fig. 2.2 shows an illustration of the their strategy.

Taking advantage of problem sparsity directly is an alternative to the explicit submapping techniques described above. The idea of explicitly targeting sparsity in the problem structure goes back to work on representing the filtered solution to SLAM as a sparse graphical model. The *Thin Junction Tree Filter* [66] sparsified the posterior by approximating it as a tree structured distribution. The (SEIF) [51] offered an alternative technique based on enforcing sparsity in the information matrix of the posterior.

Dellaert and Kaess [67] explored the use of sparse matrix factorization directly on the smoothing problem instead of the filtering approach of the EKF. By not marginalizing out any variables from the past, this approach maintains sparsity without any approximations. Aside from these contributions, the authors also explore the connection between graphical

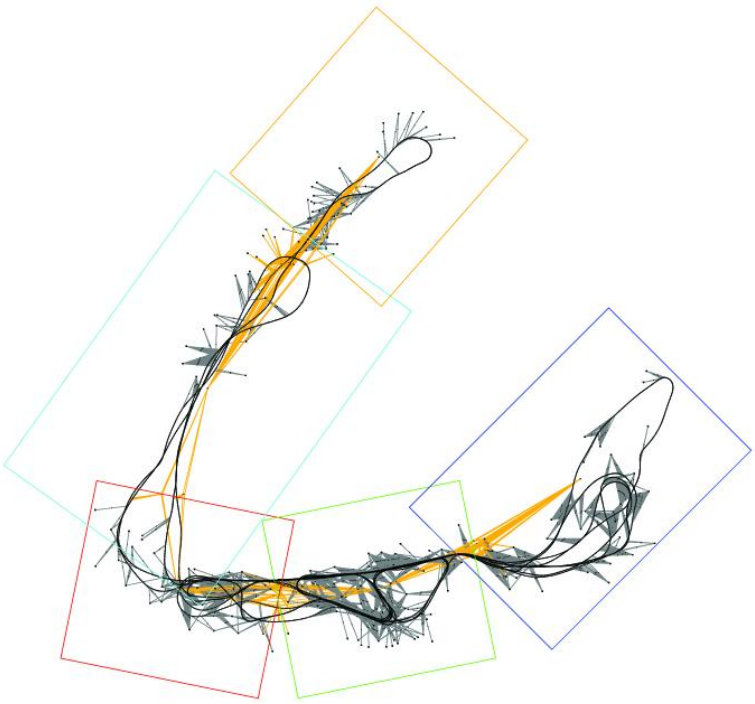


Figure 2.2: A figure from Ni et al. [2] illustrating the submapping procedure and dependencies between variables within the submaps

representations of the mapping problem and sparse matrix factorization. They note that the column ordering heuristics typically employed by sparse matrix factorization algorithms are analogous to graph triangulation techniques used in the graphical model community to construct junction trees. Along these lines Krauthausen et al. [68] investigates the used of Nested Dissection as a column order scheme and Ni and Dellaert [69] shows how the Nested Dissection scheme can be used to automatically partition a problem into recursive submaps.

Ranganathan et al. [3] uses Loopy Belief Propagation [70] to take advantage of the natural sparsity of SLAM. The information form of the problem is used decompose the optimization into individual constraints corresponds to potentials a graphical model. The problem can then be solved efficiently using message passing. By stopping the algorithm when the influence of the messages becomes negligible, Ranganathan et al. are able to dynamically adjust the amount of optimization that takes place when new data is observed, reusing as much of the old solution as possible. This allows their SLAM algorithm to run

in $O(1)$ time without loop closure, and in $O(n)$ time when loops closures occur (where n is the size of the loop).

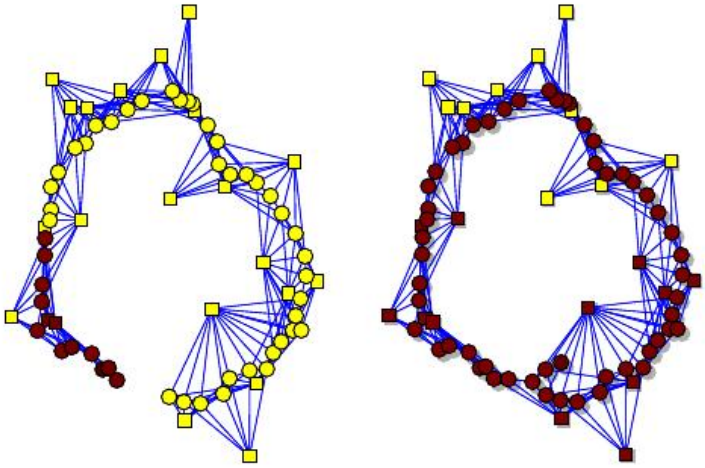


Figure 2.3: A figure from Ranganathan et al. [3] showing the operation of the algorithm when new information is added. Red colored nodes in the graph indicate variables which are being optimized, while yellow nodes are unaffected. When loop closure does not occur (left), only a small window of variables around the new information is affected. When loop closure happens (right), all of the nodes in the loop are optimized in order to spread out error along the loop as best as possible.

Kaess et al. [71] presents an incremental approach to the smoothing problem based on the connections between probabilistic graphical models and sparse matrix factorization presented in Dellaert and Kaess [67]. Kaess et al. [72] builds on this by adding a specialized data structure referred to as a *Bayes Tree*. The *Bayes Tree* allows an online approach to smoothing with incremental variable re-ordering and linearization. This eliminates the need for the batch operations required in Kaess et al. [71].

Sliding Window Filters (SWF), introduced to SLAM by Sibley et al. [73], offer a compromise between the online approach of filtering and the batch processing used in SfM. Instead of processing observations one at a time, a limited set of the most recent observations (those within the sliding window) is processed in batch mode, while all observations beyond the edge of this window are summarized with a recursively updated state estimate. The filtering disribution of a SWF of length κ is $P(x_{t-\kappa:t}|z_{1:t})$. Using such a filter allows us to improve our estimate of $x_{t-\kappa}$ using information from the future ($z_{t-\kappa+1:t}$). This

idea is particularly interesting when combined with the Iterated EKF, where it allows the linearization point of the last κ states to be iteratively updated.

Using the result of Bell and Cathey [16], which showed that the Iterated EKF is equivalent to the Gauss-Newton method for non-linear least squares, we can see that adjusting the value of κ can produce a range of algorithms from bundle-adjustment to the Kalman Filter. Setting $\kappa = t$ corresponds to solving the full least squares problem. Setting $\kappa = 1$ corresponds to the Iterated EKF. For values in between the two extremes, we are performing a full optimization over the set of variables $x_{t-\kappa:t}$, but marginalizing out $x_{t-\kappa}$ before we move on to the next time step.

2.5 Model Selection

Model selection is the primary topic of this dissertation, so in this section we review previous work from this perspective. Although it is not always explicitly called model selection, discrete decision making is nonetheless present in much of the previous work on tracking and mapping. Data association is the main model selection topic of interest and is generally acknowledged as one of the most challenging aspects of multi-target tracking. A second area of research has been on maneuvering targets where multiple distinct models are used to explain the observed behavior of a target. Since previous work on both of these topics has already been discussed in Section 2.2, this section will focus on mapping.

2.5.1 Mapping and Data Association

Unlike in tracking, data association for mapping problems is easier to circumvent with heuristics. The goal in the mapping problem is to build a reconstruction of the environment which can be used to accurately localize the robot or provide a high level map. Since the goal is not to track every single possible landmark, heuristic algorithms can be used to choose distinctive features and landmarks which are easily identifiable. This approach is particularly attractive in visual SLAM and Structure-from-Motion due to the availability of

good feature descriptors such as SIFT [55].

Agarwal et al. [1] uses SIFT to extract distinct image features and computes potential matches in a second image using an efficient approximate nearest neighbor algorithm. The candidate matches are then filtered based on a heuristic test and the RANSAC [74] algorithm is applied to the remaining matches in order to enforce geometric consistency between the two images. This procedure, which has a history in cartography dating to at least 1981 [74], transforms the data association problem into one of outlier rejection. Once a large enough list of potential feature matches is computed, RANSAC is used to select a subset of inliers. The result is used as input for an optimization procedure which takes into account all feature matches to estimate 3D structure and camera locations.

The Visual SLAM algorithm of Davison and Murray [47] uses a combination of gating and a visual search procedure for data association. The estimated landmark position and covariance from the current state of the filter are used to physically point the camera in the direction of the landmark. A 2D search is conducted to find an image patch visually similar to the original patch used to initiate the landmark. The search is restricted to the area of the image where the landmark are expected based on the filter covariance. When a match is found, it is incorporated into the filter as an observation of the landmark. The MonoSLAM system of Davison et al. [56] uses a similar approach for active feature search, except without a physically actuated camera.

Pose graph optimization approaches make use of various pre-processing strategies to establish the graph structure used in the optimization. If visual information is available, visual similarity can be used to find loop closures [75]. The precise offset between the poses can be computed using feature matching and RANSAC. With laser data, scan matching techniques such as the ICP algorithm [50] are used to compute offsets between poses in the graph.

Because laser and sonar based sensor do not provide unique identifying information about each feature, early SLAM systems based on these sensors used data association

techniques from the tracking literature combined with heuristics to identify features such as corners. Data association techniques used include gating [41, 42, 31] and MHT [76, 77] based approaches. We note that Cox and Leonard [77] recognized the fundamental similarity of multi-target tracking and SLAM when viewed from the MHT perspective; the difference being that SLAM assumes static landmarks whereas multi-target tracking assumes a static sensor.

Other ideas based on a hypothesis tree similar to the MHT have been suggested in the literature. The most relevant to this dissertation is the Joint Compatibility Branch and Bound (JCBB) [78] data association algorithm. JCBB builds a data association hypothesis tree for each measurement within one time slice. Each individual measurement z_{ti} can be assigned to any of the existing targets and the decision tree of possible assignments defines an interpretation tree. JCBB applies a branch and bound search procedure to this interpretation tree in order to find a valid assignment hypothesis which maximizes the number of matches. A hypothesis is considered valid if it passes a multivariate gating criterion (the joint compatibility test) on *all* target-measurement pairs present in the hypothesis. By gating the entire hypothesis rather than individual landmarks, the correlations between landmark estimates typical in SLAM is taken into account. A procedure similar in spirit is proposed in Hähnel et al. [79], but in this case the emphasis is on allowing data associations from the past to be reconsidered.

Algorithms based on Expectation Maximization (EM) provide an efficient alternative to explicitly considering the tree of all possible model combinations. Under this heading we consider both true EM algorithms which optimize a variational bound and EM-like algorithms which operate on intuitively similar principles. An MCMC-based algorithm for data association using EM was proposed for SfM applications in Dellaert et al. [80]. This is similar to the JPDAF [29], except the full posterior over data associations at each t is sampled with an MCMC algorithm to obtain the data association weights. Bibby and Reid [4] use classification EM within a sliding window filter to allow reversible data association and model selection between stationary and moving landmarks/targets. The dynamic

environment aspects of this paper are also discussed in the follow subsection. The Switchable Constraint [81] algorithm relaxes discrete variables into continuous variables and adds them directly into the non-linear optimization problem. The resulting optimization problem consists of weighted objectives of the form $w \cdot g(x)$ which are optimized over both w and x. This applies mainly to robust estimation problems where down-weighting the objectives can be used to discard outliers. In the same vein, the older robust estimation methods such as Iteratively Reweighted Least Squares and Huber/Cauchy kernels [82] have also been applied to mapping [83].

2.5.2 Mapping in Dynamic Environments

Solving the SLAM problem in a non-static environment requires dealing with the possibility of moving objects such as people, cars, or other robots. These challenges have given rise to many specialized algorithms which can easily be interpreted as instances of model selection. The most basic strategy is to ignore dynamic objects entirely by treating them as outliers – this is the choice implicitly made when dynamic elements of the scene are not considered. A more sophisticated strategy is to treat the two problems as roughly independent. For example, Schulz et al. [84], Hähnel et al. [85], Schulz et al. [86] are a series of results which combine a multi-target tracking system with SLAM to filter out dynamic objects before the mapping process begins. They combine a Particle Filter with the JPDAF to track people using features extracted from 2D laser range data. The resulting tracks are then used to as part of the map building procedure in order to filter out spurious data.

Wolf and Sukhatme [87] use an approach which is similar in spirit. Two occupancy maps are kept to deal with dynamic objects. The static map keeps track of stationary objects in the world and the dynamic map keeps track of the regions currently occupied by moving objects. Both maps are updated probabilistically based on a sensor model as well as a model for how each map changes in time.

In Wang et al. [88], the EKF is used for both tracking and mapping. The dependency

between object motion and robot motion is ignored in order to simplify the estimation problem and to prevent the noisy tracking data from effect the robot localization estimates. The MHT[34] is used for associating observations with landmarks and dynamic targets.

To extract more out of the dependency between the static and dynamic aspects of the scene, EM-style algorithms can be used which alternate between updating the static map and dynamic object until the two converge to a consistent interpretation. Within this category, Hähnel et al. [89] uses an EM procedure to filter out dynamic objects, effectively treating them as outliers while estimating the map. No appearance information is used to identify dynamic objects and no tracking is performed. Measurements are identified as coming from a dynamic object solely based on their inconsistency with the static map.

The SLAM-IDE[4] algorithm is also based on an EM strategy. Discrete decision variables are used to model both data association and static/dynamic classification of each observed feature. Generalized Expectation Maximization is used within a sliding window to compute an estimate for both the discrete and continuous variables involved. The sliding window approach allows data association in the past to be adjusted as more information becomes available. This is particularly important for identifying dynamic objects since motion needs to be observed over a long enough time scale to distinguish it from noise. For each landmark in the map SLAM-IDE estimates a solution under both the dynamic and static assumption, as well as the *a posteriori* probabilities for each of the two cases. Fig. 2.4 shows one time slice of the graphical model used.

More intricate relationships in the data can also be exploited at a systems level without an explicit probabilistic relationship between the static and dynamic data. This strategy is used by Leibe et al. [90] for scene modeling from a moving vehicle. Their system combines SfM estimates of the vehicle pose, discriminative detection of pedestrians/vehicles, and tracking into a single system. To track dynamic objects, they perform a custom optimization within a sliding window. For each potential object hypothesis, H, a search is performed in space-time based on the maximum-velocity of the object under its motion

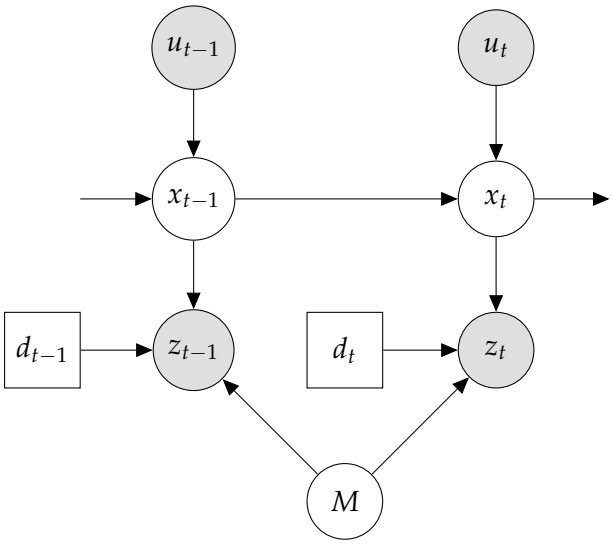


Figure 2.4: The graphical model used in SLAM-IDE. The d_t decision variable determines whether a particular landmark is stationary or dynamic. u_t denotes the control inputs to the system. The remaining notation is consistent with our previous definitions: z_t represents observations, x_t vehicle pose, and M the map.

model. Every other candidate detection which can possibly be reached from H is then included as part of the associated trajectory hypotheses. Since this search is performed independently starting from each detection within the window, each detection can be part of multiple trajectory hypotheses. Quadratic Boolean Optimization (QBO) is then used to find the optimal set of consistent trajectories subject to the constraint that no two objects can occupy the same space at the same time. The QBO problem takes into account the object motion and appearance models when linking together detections.

3

Background

This chapter provides technical background and introduces the notation which will be used for the remainder of the thesis. The mathematical notation is introduced in Section 3.1; following this we provide a brief summary of the typical tracking and mapping problem. These are specified as objectives to be minimized from the perspective of non-linear optimization rather than the equivalent maximization of log-likelihood.

Section 3.4 describes the Gauss-Newton algorithm as a generic non-linear optimization procedure applicable to both problems. In order for the Gauss-Newton algorithm to be applicable to realistic problems, sparse numerical techniques must be used. Various methods for taking advantage of sparsity are described in Section 3.5 as well the relationship between sparse estimation and the traditional filtering frameworks used for tracking.

The topic of sparsity naturally leads to a discussion of Gaussian Belief Propagation (GaBP) as a general framework for taking advantage of sparsity in estimation problems. Section 3.6 introduces the idea of Gaussian Belief Propagation on Markov Chains, where the resulting algorithms are equivalent to the Kalman Filter and Kalman Smoother. Following this, Section 3.7 describes the algorithms necessary to generalize GaBP to arbitrary graphical models.

While there is no novel material presented, we believe there is a gap in the literature of

graphical models concerning practical algorithms for generating the commonly referenced clique trees. As part of an effort to make this thesis self contained, we have provided a more explicit summary of the relevant algorithms. The majority of this chapter can be skimmed by a reader already familiar with graphical models, but we would focus attention on the notation for representing potentials and their associated operations as this will be used throughout the thesis. In particular, the notation for hybrid discrete/continuous potentials introduced in Section 3.8 is somewhat unusual, but very useful.

3.1 Basic Notation

This section introduces notation which will be used throughout the thesis. The primary objects being dealt with can be split up into two categories: observations and the state of the system being modeled. Observations will generally be denoted as Z_t, with t denoting a time index. When multiple observations are present, Z_t will additionally be split as $Z_t = (z_{t1}, \ldots, z_{t|Z_t|})$ with $|Z_t|$ separate measurements.

The generic system state at time t will be denoted as x_t. For the purposes of mapping, the system state can be split into two components, the time-dependent x_t and the global map denoted by $M = (m_1, \ldots, m_{|M|})$ when $|M|$ map elements are present. If multiple independent states are being tracked, such as in multi-target tracking, the time-dependent state will be split into multiple components: $x_t = (x_{t0}, \ldots, x_{tN})$ for N different dynamic targets plus one robot. In this case, the robot state will be represented by x_{t0}. When discrete variables are modeled as part of the system state, they will be denoted as d_t for the discrete modeling variable corresponding to x_t.

Data association variables will be denoted by Θ for track-oriented approach (associating an observation with each landmark/track) and Λ for the measurement-oriented approach (associating a landmark/track for each observation) Θ_t will denote the data associations at time t and $\Theta_{ti} = j$ will correspond to the i^{th} target, landmark or map element being associated with measurement z_{tj}. Analogously, $\Lambda_{tj} = i$ will indicate that measurement z_{tj}

was caused by the i^{th} target, landmark, or map element.

3.2 The Mapping Problem

The goal of mapping is to estimate the structure of the environment without knowing the trajectory of the robot or sensor platform. This goal is often formulated as an optimization problem over two sets of variables: the trajectory of the robot and the location of landmarks in the environment. The robot trajectory is represented by the sequence of variables representing the pose at each point in time $(p_1, \ldots, p_T)$ and the map by $M = (x_1, \ldots, x_{|M|})$. These variables are linked together through two types of objective functions. The motion model $h_{\text{mot}}(p_t)$ predicts the location at $t+1$ based on the robot pose and state at t. The observation model, $h_{\text{obs}}(p_t, x_i)$, predicts the expected sensor observation when landmark x_i is observed while the robot is in state p_t. If we consider an idealized scenario where the data association and number of landmarks is known a priori, estimation reduces to the following non-linear least squares problem

$$
\begin{aligned}
&g(p_2, \ldots, p_T, x_1, \ldots, x_{|M|}) = \\
&\sum_{t=1}^{T-1} \frac{1}{2} \| h_{\text{mot}}(p_t) - p_{t+1} \|^2_{\Sigma_{\text{mot}}} + \sum_{t=1}^{T} \sum_{i|\Theta_{ti}>0} \frac{1}{2} \| h_{\text{obs}}(p_t, x_i) - z_{t,\Theta_{ti}} \|^2_{\Sigma_{\text{obs}}} = \qquad (3.1) \\
&\sum_{t=1}^{T-1} g^{\text{mot}}_{t,t+1}(p_t, p_{t+1}) + \sum_{t=1}^{T} \sum_{i|\Theta_{ti}>0} g^{\text{obs}}_{ti}(p_t, x_i) \qquad (3.2)
\end{aligned}
$$

In the definition above, Σ_{mot} and Σ_{obs} represent the respective covariances associated with the motion and observation models. The initial robot position x_1 is held constant. In practice we cannot assume that we know the number of landmarks and the true data association, so selecting good algorithms and heuristics for data association and map management is extremely important.

Fig. 3.1 uses a factor graph to illustrate the structure of the optimization objective in Eq. 3.2 for a small toy problem. A factor graph represents all the variables involved in the problem as circular nodes and the individual local objectives as black squares (factors) labeled with the function name. An edge connects a variable with a factor whenever the

variable appears in the factor's domain.

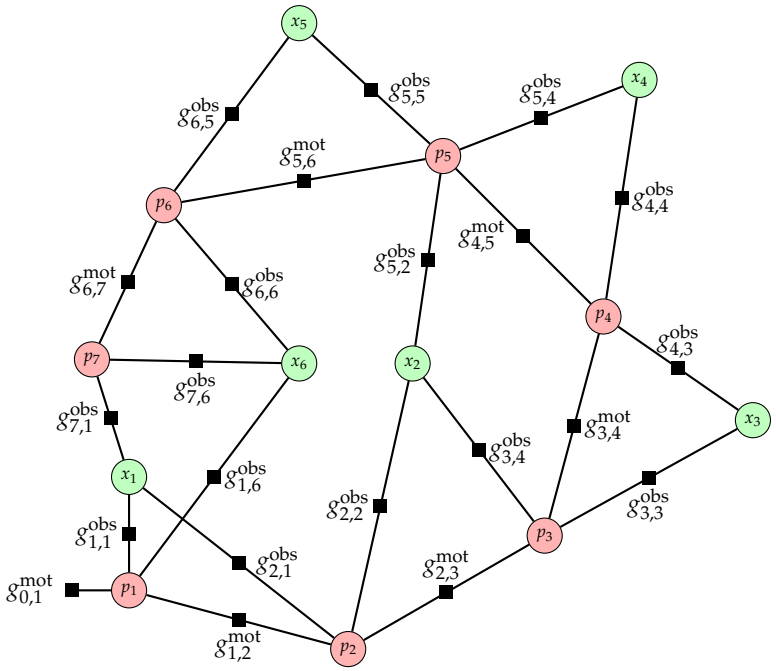


Figure 3.1: Factor graph of a sample SLAM estimation problem with $T = 7$ poses and $M = 6$ landmarks corresponding to Eq. 3.2. Red nodes represent robot poses and green nodes represent the landmarks.

Pose graph estimation is an alternative formulation of the mapping problem which avoids dealing with individual landmarks and instead solves an optimization problem over only the robot pose variables $p_{1:T}$. In this case the motion model and observation model are combined into a single objective $h_{s,t}(p_s, p_t)$ which encodes an offset between the two poses and penalizes deviation from this offset according to a metric defined by the covariance $\Sigma_{s,t}$. For constraints between two consecutive poses the objective $h_{t-1,t}$ encodes a transformation estimated from a combination of odometry and sensor data. Constraints between non-consecutive poses are generated whenever the robot revisits a location and are referred to as loop closures. The set of loop closures will be referred to as $\mathcal{L} = \{(s,t) \mid \exists h_{s,t}(p_s, p_t)\}$.

The optimization objective for pose graph SLAM is a sum over all pairwise pose constraints

$$g(p_2, \ldots, p_T) = \sum_{t=1}^{T-1} \frac{1}{2} \|h_{t,t+1}(p_t, p_{t+1})\|^2_{\Sigma_{t,t+1}} + \sum_{(t_1,t_2)\in\mathcal{L}} \frac{1}{2} \|h_{t_1,t_2}(p_{t_1}, p_{t_2})\|^2_{\Sigma_{t_1,t_2}} = \tag{3.3}$$

$$= \sum_{t=1}^{T-1} g_{t,t+1}(p_t, p_{t+1}) + \sum_{(t_1,t_2)\in\mathcal{L}} g_{t_1,t_2}(p_{t_1}, p_{t_2}) \tag{3.4}$$

The factor graph associated with this objective is shown in Fig. 3.2.

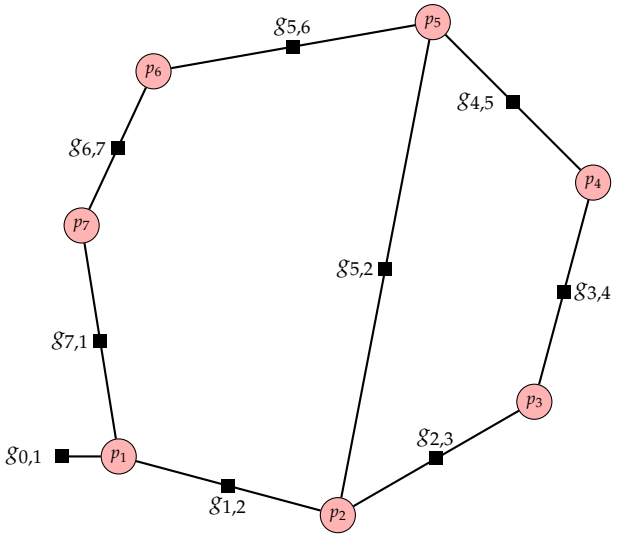


Figure 3.2: Factor graph of the pose graph estimation problem corresponding to Eq. 3.4. The robot trajectory is identical to Fig. 3.1, but all sensor data is incorporated using pairwise relationships between the poses.

3.3 The Tracking Problem

Tracking can be thought of as a special case of the mapping problem where the landmarks move and the sensor is stationary. Despite this, there are several important differences between the two which cause the estimation algorithms to differ substantially in practice. One major distinction is that data association can be somewhat side stepped in the mapping problem. This is due to the fact that we do not need to recover every possible landmark in the scene – only enough landmarks to reconstruct the trajectory accurately. This gives some leeway in picking easier landmarks to track which is not present in the tracking problem. For tracking, the primary focus is on the trajectories of the targets and so data

association takes a much more prominent role. That said, if we assume data association is known in advance, the estimation problem can also be expressed as sparse non-linear least squares. To do we will reuse the notation of the mapping section above as much as possible. Assuming N targets are being tracked, the states of the targets will be denoted by $\{x_{t1}, \ldots, x_{tN}\}$. As before, Θ_{ti} will provide the data association with $\Theta_{ti} = j$ indicating z_{tj} is an observation of the i$^{\text{th}}$ target x_{ti}.

The motion model for each target is analogous to the motion model of the robot/sensor in Section 3.2, and the observation model corresponds directly to landmark observations in that section. The overall objective for the problem is

$$g(p_2, \ldots, p_T, x_1, \ldots, x_T) =$$

$$= \sum_{t=1}^{T-1} \sum_{i=0}^{N} \frac{1}{2} \| h_{\text{mot}}(x_{t,i}) - x_{t+1,i} \|^2_{\Sigma_{\text{mot}}} + \sum_{t=1}^{T} \sum_{i:\Theta_{ti}>0} \frac{1}{2} \| h_{\text{obs}}(x_{t,i}) - z_{t,\Theta_{ti}} \|^2_{\Sigma_{\text{obs}}} = \tag{3.5}$$

$$= \sum_{t=1}^{T-1} \sum_{i=0}^{N} g^{\text{mot}}_{t,i}(x_{t,i}, x_{t+1,i}) + \sum_{t=1}^{T} \sum_{i:\Theta_{ti}>0} g^{\text{obs}}_{t,i}(x_{t,i}) \tag{3.6}$$

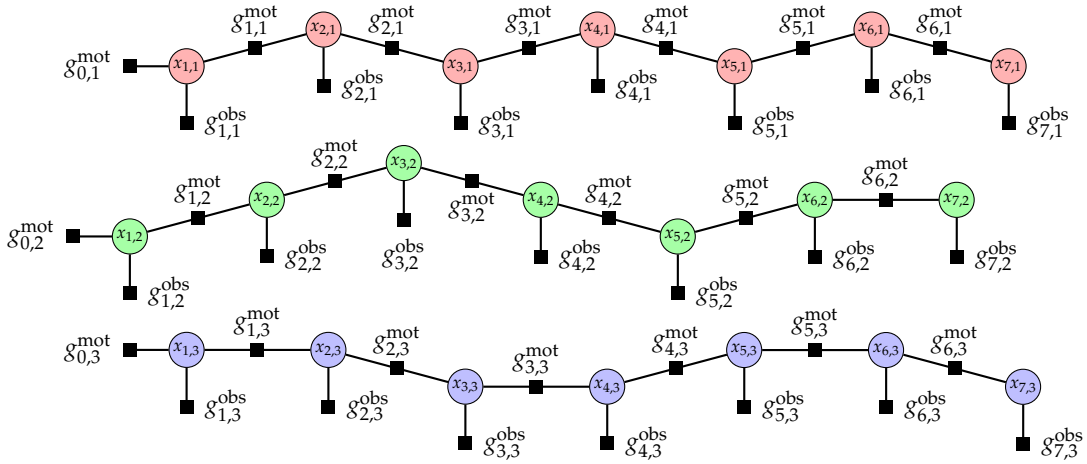


Figure 3.3: Factor graph of a tracking problem with known data association, $T = 7$, and $N = 3$ targets corresponding to Eq. 3.6.

3.4 The Gauss-Newton Method

The sections above defined the tracking and mapping problems as non-linear least squares optimization problems. The Gauss-Newton method is a standard algorithm for the prob-

lems defined by Eq. 3.1, Eq. 3.3, and Eq. 3.5. Given a set of objectives $g_i(x_i) : \mathbb{R}^{n_i} \to \mathbb{R}$, with x_i a (possibly overlapping) subsets of the problem domain $x \in \mathbb{R}^n$, the goal is to find a value for the state vector x which minimizes the combined objective

$$g = \sum_i g_i(x_i) \tag{3.7}$$

We will use the index i to index the objective g_i as well as the domain of g_i within the global state vector x. In this notation, x_i refers to the elements of x over which g_i is defined. For some situations it will be convenient to use a double index for the set of objectives, i.e. g_{ij}. In this case the corresponding domain will be referred to as x_{ij}. To simplify notation, we define the addition of such sub-vectors so that $x_i + x_j$ is the sum over corresponding indices. The missing entries in each of the two vectors are treated as zeros; e.g. we are treating each x_i as a sparse vector in the full problem domain $\mathbb{R}^n$. The same convention will be used with the gradient vectors and Hessian matrices corresponding to each sub-domain.

The individual g_i terms in Eq. 3.7 can be expanded to make the least-squares formulation explicit

$$g_i(x_i) = \frac{1}{2} f_i(x_i)^\top f_i(x_i) \tag{3.8}$$

with $f_i : \mathbb{R}^{n_i} \to \mathbb{R}^{m_i}$. If we linearize f_i at x^*

$$\tilde{f}_i^{[x_i^*]}(x_i) = f_i(x^*) + \nabla f_i(x_i^*)^\top (x_i - x_i^*) \tag{3.9}$$

$$= f_i(x^*) + \nabla f_i(x_i^*)^\top \delta_i \tag{3.10}$$

we get the classic Gauss-Newton minimization scheme which iteratively minimizes the approximate linear least-squares objective

$$\tilde{g}^{[x^*]}(x) = \sum_i \tilde{g}_i^{[x^*]}(x_i) = \sum_i \frac{1}{2} \tilde{f}_i^{[x^*]}(x_i)^\top \tilde{f}_i^{[x^*]}(x_i) \tag{3.11}$$

Expanding each term in Eq. 3.11 results in the following quadratic form

$$\begin{aligned} \tilde{g}^{[x^*]}(x) =& \sum_i \left(\frac{1}{2} f_i(x^*)^\top f_i(x^*) + f_i(x^*)^\top \nabla f_i(x_i^*)^\top \delta_i + \frac{1}{2} \delta_i^\top \nabla f_i(x_i^*) \nabla f_i(x_i^*)^\top \delta_i \right) \\ =& \frac{1}{2} f(x^*) + \mathbf{g}\mathbf{r}^\top \delta + \frac{1}{2} \delta^\top \mathbf{H} \delta \end{aligned} \tag{3.12}$$

By setting the derivative of Eq. 3.12 to 0, we can compute the optimal δ and use it to update the linearization point

$$\mathbf{gr} = \sum_i \mathbf{gr}_i = \sum_i \nabla f_i(x_i^*)^\top f_i(x_i^*) \tag{3.13}$$

$$\mathbf{H} = \sum_i \mathbf{H}_i = \sum_i \nabla f_i(x_i^*)^\top \nabla f_i(x_i^*) \tag{3.14}$$

$$\delta = -\mathbf{H}^{-1}\mathbf{gr} \tag{3.15}$$

$$x^*_{\text{new}} = x^* + \delta \tag{3.16}$$

The Gauss-Newton algorithm proceeds by repeating the linearization and quadratic minimization procedure until convergence.

3.5 Sparse Optimization and the Schur Compliment

For large systems of small objectives (i.e. $n_i \ll n$), the quadratic minimization problem in Eq. 3.11 is often sparse. This sparsity allows Eq. 3.15 to be scaled to problems with thousands of variables while still maintaining reasonable performance. There are several common strategies to do this – some fairly general and some problem specific. To talk about these approaches it is helpful to visualize the sparsity pattern of the Hessian matrix H. The sparsity patterns of the Hessians associated with the mapping and tracking optimization problems are illustrated in Fig. 3.4 and Fig. 3.5. These patterns match the connectivity of the associated factor graphs (Fig. 3.1, Fig. 3.2, and Fig. 3.3). Since each factor denotes a relationship between the associated variables, it implies a potential non-zero entry in the Hessian matrix.

For the SLAM and SfM problems illustrated in Fig. 3.1, block matrix inversion is a common problem-specific sparsity strategy for solving Eq. 3.15. Looking at Fig. 3.4a, we can partition the Hessian and gradient into blocks corresponding to the pose and landmark variables, indicated by the **p** and **x** subscripts respectively.

$$\begin{bmatrix} \mathbf{H_{pp}} & \mathbf{H_{px}} \\ \mathbf{H_{xp}} & \mathbf{H_{xx}} \end{bmatrix} \begin{bmatrix} \delta_p \\ \delta_x \end{bmatrix} = \begin{bmatrix} \mathbf{gr_p} \\ \mathbf{gr_x} \end{bmatrix} \tag{3.17}$$

The decomposition in Eq. 3.17 can be used to take advantage of sparsity by applying Gaussian elimination to the blocks (see Schur compliment, Boyd and Vandenberghe [91]). To do so solve we first symbolically solve for δ_x in terms of δ_p using the second block-row.

$$\delta_x = \mathbf{H}_{\mathbf{xx}}^{-1}\Big(\mathbf{gr}_{\mathbf{x}} - \mathbf{H}_{\mathbf{xp}}\delta_p\Big) \tag{3.18}$$

The resulting expression is substituted into the second block-row to obtain an equivalent system in terms of only the x_2 variables.

$$S = \mathbf{H}_{\mathbf{pp}} - \mathbf{H}_{\mathbf{px}}\mathbf{H}_{\mathbf{xx}}^{-1}\mathbf{H}_{\mathbf{xp}} \tag{3.19}$$

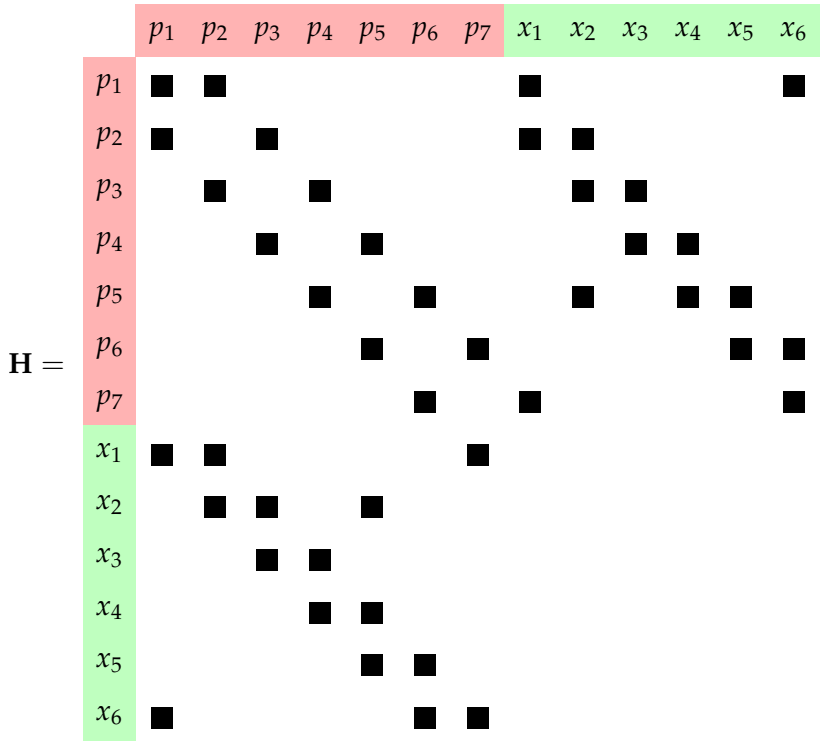
$$\delta_p = S^{-1}\Big(\mathbf{gr}_{\mathbf{p}} - \mathbf{H}_{\mathbf{px}}\mathbf{H}_{\mathbf{xx}}^{-1}\mathbf{gr}_{\mathbf{x}}\Big) \tag{3.20}$$

In the above equation S is known as the Schur compliment of $\mathbf{H}_{\mathbf{pp}}$. Once δ_p is known, the corresponding value of δ_x can be computed directly from Eq. 3.18 via back substitution.

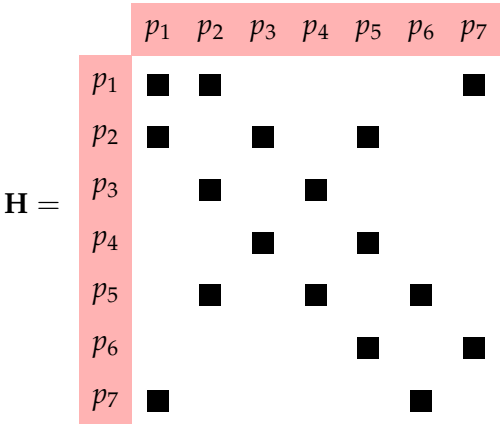
The method described above does not reduce computational complexity for dense matrices, but it can help when the Hessian is sparse. Two matrix inverses are needed: $\mathbf{H}_{\mathbf{xx}}^{-1}$ and S^{-1}. In Fig. 3.4a we can see that $\mathbf{H}_{\mathbf{xx}}$ is a diagonal matrix whose inversion is trivial. On the other hand, S is of the same size as $\mathbf{H}_{\mathbf{pp}}$ and computing S^{-1} is a relatively small problem because there are typically far fewer poses than landmarks. This technique is often referred to as the Schur compliment trick and has a long history in the literature [92].

More general sparse matrix decomposition algorithms can take advantage of arbitrary matrix sparsity when solving the quadratic optimization problem [93]. These techniques are also applicable to pose graph estimation, where there is no guarantee that any particular submatrix has diagonal or block diagonal structure.

For the special case of multi-target tracking, a Kalman filter is often used for each of the trajectories shown in Fig. 3.5. The Kalman filter can also be thought of as a sparse linear-least squares algorithm [16, 18] and at its core also relies on iterative application of the Schur compliment trick. The following section will illustrate that Gaussian Graphical Models can be thought of as a generalization of this idea to arbitrary sparse problems.



(a) SLAM/SfM formulation of the mapping problem



(b) Pose graph formulation of the mapping problem

Figure 3.4: Typical sparsity patterns for mapping problems.

$$\mathbf{H} = \begin{bmatrix} H_1 & 0 & 0 \\ 0 & H_2 & 0 \\ 0 & 0 & H_3 \end{bmatrix}$$

$$\mathbf{H_1} = \begin{array}{c|ccccccc} & x_{11} & x_{21} & x_{31} & x_{41} & x_{51} & x_{61} & x_{71} \\ \hline x_{11} & \blacksquare & \blacksquare & & & & & \\ x_{21} & \blacksquare & \blacksquare & \blacksquare & & & & \\ x_{31} & & \blacksquare & \blacksquare & \blacksquare & & & \\ x_{41} & & & \blacksquare & \blacksquare & \blacksquare & & \\ x_{51} & & & & \blacksquare & \blacksquare & \blacksquare & \\ x_{61} & & & & & \blacksquare & \blacksquare & \blacksquare \\ x_{71} & & & & & & \blacksquare & \blacksquare \end{array}$$

$$\mathbf{H_2} = \begin{array}{c|ccccccc} & x_{12} & x_{22} & x_{32} & x_{42} & x_{52} & x_{62} & x_{72} \\ \hline x_{12} & \blacksquare & \blacksquare & & & & & \\ x_{22} & \blacksquare & \blacksquare & \blacksquare & & & & \\ x_{32} & & \blacksquare & \blacksquare & \blacksquare & & & \\ x_{42} & & & \blacksquare & \blacksquare & \blacksquare & & \\ x_{52} & & & & \blacksquare & \blacksquare & \blacksquare & \\ x_{62} & & & & & \blacksquare & \blacksquare & \blacksquare \\ x_{72} & & & & & & \blacksquare & \blacksquare \end{array}$$

$$\mathbf{H_3} = \begin{array}{c|ccccccc} & x_{13} & x_{23} & x_{33} & x_{43} & x_{53} & x_{63} & x_{73} \\ \hline x_{13} & \blacksquare & \blacksquare & & & & & \\ x_{23} & \blacksquare & \blacksquare & \blacksquare & & & & \\ x_{33} & & \blacksquare & \blacksquare & \blacksquare & & & \\ x_{43} & & & \blacksquare & \blacksquare & \blacksquare & & \\ x_{53} & & & & \blacksquare & \blacksquare & \blacksquare & \\ x_{63} & & & & & \blacksquare & \blacksquare & \blacksquare \\ x_{73} & & & & & & \blacksquare & \blacksquare \end{array}$$

Figure 3.5: Typical sparsity pattern for multi-target tracking problems. Note that the estimation problem for each track is in fact completely independent. While it is not common to think of this as problem sparsity, it is in fact an extreme example where the estimation for each track is completely independent once data association is established.

3.6 Gaussian Belief Propagation: Markov Chains

Gaussian Belief Propagation is a probabilistic alternative to the linear algebra based approaches to sparsity discussed in the previous section. Since maximum likelihood inference in a linear Gaussian model is equivalent to linear least-squares, we can switch to a probabilistic perspective in order to take advantage of well developed sparse inference algorithms. Gaussian Belief Propagation is an application of Belief Propagation to Gaussian graphical models [94, 95]. The resulting algorithm is very similar to the block Gaussian elimination techniques described in the previous section. This connection to graphical models and general inference techniques motivates the inclusion of discrete variables in Chapter 4. A detailed presentation of these topics is beyond the scope of this thesis, but we will provide an overview geared towards defining the notation and providing a reference for the remaining chapters.

3.6.1 Belief Propagation and Message Passing

The Belief Propagation algorithm is a very general application of dynamic programming to sparse inference. It is applicable to both MAP inference and the computation of posterior distributions. We use a simple estimation problem as an example to introduce the topic at an intuitive level. The problem consists of estimating the state of a system of three variables $(x_1, x_2, x_3) \in \mathbb{R}^3$ from the corresponding observations (z_1, z_2, z_3). The posterior for this problem is proportional to the following product

$$\mathbf{P}(x_{1:3} \mid z_{1:3}) \propto \underbrace{\mathbf{P}(z_1 \mid x_1)\,\mathbf{P}(x_1)}_{\mathbf{L}(x_1)}\,\underbrace{\mathbf{P}(z_2 \mid x_2)\,\mathbf{P}(x_2 \mid x_1)}_{\mathbf{L}(x_1,x_2)}\,\underbrace{\mathbf{P}(z_3 \mid x_3)\,\mathbf{P}(x_3 \mid x_2)}_{\mathbf{L}(x_2,x_3)} \tag{3.21}$$

The MAP estimate is given by

$$x^*_{1:3} = \operatorname*{argmax}_{x_{1:3}} \left\{ \mathbf{L}(x_1)\,\mathbf{L}(x_1, x_2)\,\mathbf{L}(x_2, x_3) \right\} \tag{3.22}$$

Assuming all of the component distributions in Eq. 3.21 are Gaussian, we can compute $x^*_{1:3}$ by computing the parameters of the distribution $\mathbf{P}(x_{1:3} \mid z_{1:3})$ and looking at its mean.

Despite the fact that we know x_1 and x_3 do not directly interact, this information is lost once we compute $\mathbf{P}(x_1, x_2, x_3)$. As an alternative, we can incrementally compute the maximal values while reusing the partial computations as much as possible; this is the dynamic programming approach. For the example above, we can rewrite the maximization to emphasize the repetitive computations inherent in the calculation.

$$x_1^* = \underset{x_1}{\operatorname{argmax}} \left\{ \max_{x_2} \left\{ \mathbf{L}(x_1)\, \mathbf{L}(x_1, x_2) \max_{x_3} \left\{ \mathbf{L}(x_2, x_3) \right\} \right\} \right\} \tag{3.23}$$

$$x_2^* = \underset{x_2}{\operatorname{argmax}} \left\{ \max_{x_1} \left\{ \mathbf{L}(x_1)\, \mathbf{L}(x_1, x_2) \right\} \max_{x_3} \left\{ \mathbf{L}(x_2, x_3) \right\} \right\} \tag{3.24}$$

$$x_3^* = \underset{x_3}{\operatorname{argmax}} \left\{ \max_{x_2} \left\{ \max_{x_1} \left\{ \mathbf{L}(x_1)\, \mathbf{L}(x_1, x_2) \right\} \mathbf{L}(x_2, x_3) \right\} \right\} \tag{3.25}$$

The repetition becomes more obvious if we define temporary distributions $\phi_1(x_1, x_2)$, $\phi_2(x_2, x_3)$, $\overrightarrow{\mu}(x_2)$, and $\overleftarrow{\mu}(x_2)$ as follows

$$x_1^* = \underset{x_1}{\operatorname{argmax}} \left\{ \max_{x_2} \left\{ \overbrace{\mathbf{L}(x_1)\, \mathbf{L}(x_1, x_2)}^{\phi_1(x_1, x_2)} \underbrace{\max_{x_3} \left\{ \overbrace{\mathbf{L}(x_2, x_3)}^{\phi_2(x_2, x_3)} \right\}}_{\overleftarrow{\mu}(x_2)} \right\} \right\} \tag{3.26}$$

$$x_2^* = \underset{x_2}{\operatorname{argmax}} \left\{ \underbrace{\max_{x_1} \left\{ \overbrace{\mathbf{L}(x_1)\, \mathbf{L}(x_1, x_2)}^{\phi_1(x_1, x_2)} \right\}}_{\overrightarrow{\mu}(x_2)} \underbrace{\max_{x_3} \left\{ \overbrace{\mathbf{L}(x_2, x_3)}^{\phi_2(x_2, x_3)} \right\}}_{\overleftarrow{\mu}(x_2)} \right\} \tag{3.27}$$

$$x_3^* = \underset{x_3}{\operatorname{argmax}} \left\{ \max_{x_2} \left\{ \underbrace{\max_{x_1} \left\{ \overbrace{\mathbf{L}(x_1)\, \mathbf{L}(x_1, x_2)}^{\phi_1(x_1, x_2)} \right\}}_{\overrightarrow{\mu}(x_2)} \overbrace{\mathbf{L}(x_2, x_3)}^{\phi_2(x_2, x_3)} \right\} \right\} \tag{3.28}$$

The temporary distributions ϕ_1 and ϕ_2 are referred to as the local potentials; these capture the local relationships between variables. The other two temporary distributions, $\overrightarrow{\mu}$ and $\overleftarrow{\mu}$, are referred to as messages. The forward message $\overrightarrow{\mu}$ captures all information about x_2 from ϕ_1. Similarly, the backward message $\overleftarrow{\mu}$ captures all information about x_2 from ϕ_2. The messages transmit information between the two local potentials and allow a global solution to be coordinated. Belief propagation, when run on this example results in the

follow computations

$$\phi_1(x_1, x_2) := \mathbf{L}(x_1)\,\mathbf{L}(x_1, x_2) \tag{3.29}$$

$$\phi_2(x_2, x_3) := \mathbf{L}(x_2, x_3) \tag{3.30}$$

$$\overrightarrow{\mu}(x_2) := \max_{x_1} \left\{\phi_1(x_1, x_2)\right\} \tag{3.31}$$

$$\overleftarrow{\mu}(x_2) := \max_{x_3} \left\{\phi_2(x_2, x_3)\right\} \tag{3.32}$$

$$\mathcal{B}_1(x_1, x_2) := \phi_1(x_1, x_2)\,\overleftarrow{\mu}(y) \tag{3.33}$$

$$\mathcal{B}_2(x_2, x_3) := \phi_2(x_2, x_3)\,\overrightarrow{\mu}(y) \tag{3.34}$$

$$(x_1^*, x_2^*) := \operatorname*{argmax}_{x_1, x_2} \left\{\mathcal{B}_1(x_1, x_2)\right\} \tag{3.35}$$

$$(x_2^*, x_3^*) := \operatorname*{argmax}_{x_2, x_3} \left\{\mathcal{B}_1(x_1, x_2)\right\} \tag{3.36}$$

After computing the local potentials and the messages, results are stored in the 'belief' distributions $\mathcal{B}_1$ and $\mathcal{B}_2$. These combine information from the local potentials with the incoming message in order to compute a proxy for the global objective over the local variables. Note that despite the fact that x_2^* is computed twice, the optimal values will be the same as long as all of the local potentials are not degenerate distributions. This is a consequence of the fact that non-degenerate Gaussian distributions have a unique maximal likelihood value.

The intuition behind the Belief Propagation algorithm is similar to block Gaussian elimination algorithm. Instead of maximizing $\mathbf{L}(x_1, x_2, x_3)$ over (x_1, x_2, x_3), we are symbolically maximizing ϕ_1 over x_1 in closed form before combining it with ϕ_2 to compute the maximal value over (x_2, x_3). In this particular example, there is not much of a computational advantage due to the overhead of the algorithm compared to the amount of sparsity in the problem. For larger problems the improvement can be drastic. If each of x_1, x_2, and x_3 represented n-dimensional vectors instead of single variables, the algorithm above would replace a least-squares problem of size $3n$ with several problems of size $2n$. Due to the roughly $O(n^3)$ computational cost of Cholesky decomposition, this makes a larger difference as n increases.

3.6.2 Gaussian Potentials

Gaussian Belief Propagation requires manipulating various Gaussian distributions in closed form. To do so we must define data structures capable of storing the various potentials as well as the set of allowed operations on these data structures (e.g. maximization, integration, and computing products). For many of the required operations, the 'information form' of the Gaussian distribution is a much more convenient parameterization than the standard moment form. In addition, the information matrix naturally corresponds to the Hessian matrix and so directly reveals the sparsity structure of the problem. This is not the case with the covariance matrix, which is typically dense even for sparse Gaussian distributions.

The probability density function of $x \sim \mathcal{N}(\mu, \Sigma)$ is given by

$$\mathbf{P}(x \mid \mu, \Sigma) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) \tag{3.37}$$

Instead of the moments μ and Σ, the information form is defined via the information matrix I and the information vector η

$$I \equiv \Sigma^{-1} \tag{3.38}$$

$$\eta \equiv \Sigma^{-1}\mu \tag{3.39}$$

We can rewrite the density from Eq. 3.37 in information form by expanding the term inside the exponent and applying the above definitions

$$\mathbf{P}(x \mid \eta, I) = (2\pi)^{-\frac{n}{2}} |I|^{\frac{1}{2}} \exp\left(-\frac{1}{2}x^\top I x + x^\top \eta - \frac{1}{2}\eta^\top I^{-1}\eta\right) \tag{3.40}$$

We note that the log-likelihood of Eq. 3.40 is always a quadratic form in x

$$\log \mathbf{P}(x) = -\frac{n}{2}\log(2\pi) + \frac{1}{2}\log|I| + \left(-\frac{1}{2}x^\top I x + x^\top \eta - \frac{1}{2}\eta^\top I^{-1}\eta\right) \tag{3.41}$$

$$= \frac{1}{2}x^\top(-I)x + x^\top \eta + \left(\frac{1}{2}\log|I| - \frac{n}{2}\log(2\pi) - \frac{1}{2}\eta^\top I^{-1}\eta\right) \tag{3.42}$$

$$= \frac{1}{2}x^\top A x + x^\top b + c \tag{3.43}$$

This motivates the definition of a general *Gaussian Potential*, $\phi(x)$ as the exponent of a quadratic form. The Gaussian Potential $\phi : \mathbb{R}^n \to \mathbb{R}$ is represented by the three parameters A_ϕ, b_ϕ, and c_ϕ such that

$$\phi(x) \equiv \exp\left(\frac{1}{2}x^\top A_\phi x + x^\top b_\phi + c_\phi\right) \tag{3.44}$$

This definition does not always correspond to a valid Gaussian distribution, but any Gaussian distribution can be represented as above. To represent a standard multivariate Gaussian with information parameters (I, η), the potential parameters are set to

$$A_\phi = -I \tag{3.45}$$

$$b_\phi = \eta \tag{3.46}$$

$$c_\phi = \frac{1}{2}\log|I| - \frac{n}{2}\log(2\pi) - \frac{1}{2}\eta^\top I^{-1}\eta \tag{3.47}$$

Gaussian Potentials can also be used to represent conditional distributions where the mean is parameterized by an affine function

$$x \mid y \sim \mathcal{N}(\mu(y), \Sigma) \tag{3.48}$$

$$\mu(y) = Hy + \mu_0 \tag{3.49}$$

If we write the information vector, η, as a function of y and express the log-likelihood in terms of the parameters I, y, and μ_0 we get

$$\eta(y) = I(Hy + \mu_0) \tag{3.50}$$

$$\log \mathbf{P}(x \mid y) = -\frac{n}{2}\log(2\pi) + \frac{1}{2}\log|I| - \frac{1}{2}x^\top Ix + x^\top \eta(y) - \frac{1}{2}\eta(y)^\top I^{-1}\eta(y) \tag{3.51}$$

By expanding the definition of $\eta(y)$ and rearranging terms, we can rewrite Eq. 3.51 as a quadratic form of the combined vector (x, y) parametrized by I, H, and μ_0

$$\log \phi(x, y) \equiv \log \mathbf{P}(x \mid y) = \frac{1}{2}\begin{bmatrix} x \\ y \end{bmatrix}^\top \begin{bmatrix} A_{xx} & A_{xy} \\ A_{yx} & A_{yy} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}^\top \begin{bmatrix} b_x \\ b_y \end{bmatrix} + c \tag{3.52}$$

with

$$A_{xx} = -I \tag{3.53}$$

$$A_{xy} = A_{yx}^{\top} = IH \tag{3.54}$$

$$A_{yy} = -H^{\top} IH \tag{3.55}$$

$$b_x = I\mu_0 \tag{3.56}$$

$$b_y = -H^{\top} I\mu_0 \tag{3.57}$$

$$c = \frac{1}{2}\Big(\mu_0^{\top} I\mu_0 + n\log(2\pi) - \log|I|\Big) \tag{3.58}$$

The potentials are closed under multiplication, division, and exponentiation. That is, $\phi_1 * \phi_2$, ϕ_1/ϕ_2, and ϕ_1^{α} are all valid potentials. As an example, the multiplication of two potentials $\phi_1 * \phi_2$ results in the product potential ϕ_{prod} with parameters

$$A_{\phi_{\text{prod}}} = A_{\phi_1} + A_{\phi_2} \tag{3.59}$$

$$b_{\phi_{\text{prod}}} = b_{\phi_1} + b_{\phi_2} \tag{3.60}$$

$$c_{\phi_{\text{prod}}} = c_{\phi_1} + c_{\phi_2} \tag{3.61}$$

These properties are a consequence of the definition in Eq. 3.44 and the linearity of quadratic forms in terms of their parameters. When multiplying or diving two potentials which do not have the same domain, e.g. $\phi_{\text{prod}}(x,y,z) = \phi_1(x,y) \cdot \phi_2(y,z)$ the factor potentials are extended to cover the full domain (x,y,z). This is done by setting the quadratic form parameters correspond to the added variables to zero. Letting A_1, b_1, c_1 and A_2, b_2, c_2 be the parameters of ϕ_1 and ϕ_2 respectively, the product parameters are computed as

$$A_{\phi_{\text{prod}}} = \begin{bmatrix} A_{1xx} & A_{1xy} & 0 \\ A_{1yx} & A_{1yy} & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & A_{2yy} & A_{2yz} \\ 0 & A_{2zy} & A_{2zz} \end{bmatrix} \tag{3.62}$$

$$b_{\phi_{\text{prod}}} = \begin{bmatrix} b_{1x} \\ b_{1y} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ b_{2y} \\ b_{2z} \end{bmatrix} \tag{3.63}$$

$$c_{\phi_{\text{prod}}} = c_1 + c_2 \tag{3.64}$$

The operations $\phi_a \cdot \phi_b$ and ϕ_a/ϕ_b are defined in terms of the respective sum and different of the parameters of the quadratic forms.

If a potential ϕ is negative semi-definite, that is the matrix A_ϕ is negative semi-definite, we can also compute integrals and maximize in closed form. To do so for the potential $\phi(x, y)$ we partition its parameters into blocks corresponding to the $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ variables

$$A_\phi = \begin{bmatrix} A_{xx} & A_{xy} \\ A_{yx} & A_{yy} \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)} \tag{3.65}$$

$$b_\phi = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \in \mathbb{R}^{n+m} \tag{3.66}$$

Using the partitioned parameters, $\phi_{\texttt{marg}}(x) = \int \phi(x, y)\, \mathrm{d}y$ is given by

$$A_{\phi_{\text{marg}}} = A_{xx} - A_{xy} A_{yy}^{-1} A_{yx} \tag{3.67}$$

$$b_{\phi_{\text{marg}}} = b_x - A_{xy} A_{yy}^{-1} b_y \tag{3.68}$$

$$c_{\phi_{\text{marg}}} = c_\phi - \frac{1}{2} b_y^\top A_{yy}^{-1} b_y + \frac{1}{2} m \log(2\pi) - \frac{1}{2} \log\left|-A_{yy}\right| \tag{3.69}$$

Similarly, $\phi_{\texttt{max}}(x) = \max_y \left\{\phi(x, y)\right\}$ has parameters

$$A_{\phi_{\text{max}}} = A_{xx} - A_{xy} A_{yy}^{-1} A_{yx} \tag{3.70}$$

$$b_{\phi_{\text{max}}} = b_x - A_{xy} A_{yy}^{-1} b_y \tag{3.71}$$

$$c_{\phi_{\text{max}}} = c_\phi - \frac{1}{2} b_y^\top A_{yy}^{-1} b_y; \tag{3.72}$$

Note that the only difference between maximization and marginalization of Gaussian Potentials is the constant factor in the quadratic form. Maximization and marginalization (integration) of potentials will be a commonly used operation in this thesis, so we define the following operators as short hand for the operations described above.

$$\underset{y}{\mathbf{marg}}[\phi] \equiv \int \phi(x, y)\, \mathrm{d}y \tag{3.73}$$

$$\underset{y}{\mathbf{max}}[\phi] \equiv \max_y \left\{\phi(x, y)\right\} \tag{3.74}$$

These operators take a potential as input and return a modified potential representing the result of the specified operation.

Finally, it is occasionally useful to define a conditioning or partial evaluation operation for Gaussian Potentials. This comes into play when a probabilistic model needs to be

conditioned on actual observations prior to inference. Given $\phi(x, y)$ parameterized as in Eq. 3.65 and Eq. 3.66, we can define a partial evaluation operation which fixes $y = y_0$ at a single value. The potential $\phi_{\text{eval}}(x) = \phi(x, y{=}y_0)$ has parameters given by

$$A_{\phi_{\text{eval}}} = A_{xx} \tag{3.75}$$

$$b_{\phi_{\text{eval}}} = b_x + A_{xy} y_0 \tag{3.76}$$

$$c_{\phi_{\text{eval}}} = c_\phi + b_y^\top y_0 + \frac{1}{2} y_0^\top A_{yy} y_0; \tag{3.77}$$

This operation is efficient in information form as it does not require any matrix inversions. In the covariance parameterization, computing the parameters of a partial evaluation operation requires computing the Schur complement.

3.6.3 The Kalman Filter and Smoother

Using the principles from Section 3.6.1 and the operations from Section 3.6.2 we can define the Kalman Filter and Smoother as special cases of Belief Propagation. If the motion and observation models are linear with Gaussian noise, we can define local potentials for each t

$$\phi_1(x_1) \equiv \mathbf{P}(x_1)\, \mathbf{P}(z_1 \mid x_1) \tag{3.78}$$

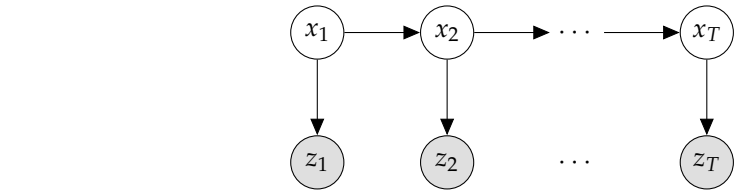
$$\phi_2(x_1, x_2) \equiv \mathbf{P}(x_2 \mid x_1)\, \mathbf{P}(z_2 \mid x_2) \tag{3.79}$$

$$\vdots$$

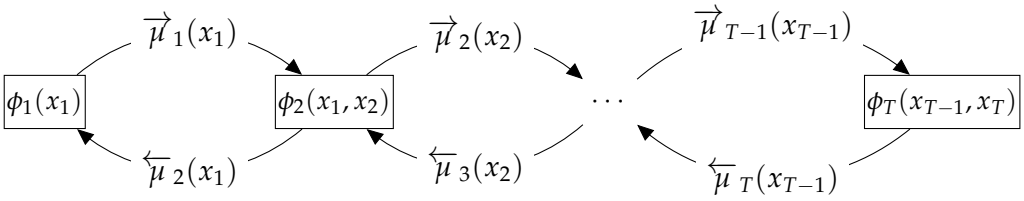
$$\phi_t(x_{t-1}, x_t) \equiv \mathbf{P}(x_t \mid x_{t-1})\, \mathbf{P}(z_t \mid x_t) \tag{3.80}$$

$$\vdots$$

Given the graphical model and clique tree shown in Fig. 3.6 combined with the potentials defined above, the message passing algorithm is composed of the following basic



(a) Bayesian Network



(b) Clique tree

Figure 3.6: Graphical model and associated clique tree for the filtering/smoothing problem.

recursions for $t = 2, \ldots, T-1$

$$\overrightarrow{\mu}_t(x_t) \coloneqq \max_{x_{t-1}} \left\{ \overrightarrow{\mu}_{t-1}(x_{t-1}) \phi_t(x_{t-1}, x_t) \right\} \quad \equiv \quad \max_{x_{t-1}} \left[\overrightarrow{\mu}_{t-1} \cdot \phi_t \right] \tag{3.81}$$

$$\overleftarrow{\mu}_t(x_{t-1}) \coloneqq \max_{x_t} \left\{ \phi_t(x_{t-1}, x_t) \overleftarrow{\mu}_{t+1}(x_t) \right\} \quad \equiv \quad \max_{x_t} \left[\phi_t \cdot \overleftarrow{\mu}_{t+1} \right] \tag{3.82}$$

$$\mathcal{B}_t(x_{t-1}, x_t) \coloneqq \overrightarrow{\mu}_{t-1}(x_{t-1}) \phi_t(x_{t-1}, x_t) \overleftarrow{\mu}_{t+1}(x_t) \quad \equiv \quad \overrightarrow{\mu}_{t-1} \cdot \phi_t \cdot \overleftarrow{\mu}_{t+1} \tag{3.83}$$

The forward recursion is specified by Eq. 3.81, the backward recursion by Eq. 3.82, and the local posterior in Eq. 3.83 combines the two. The resulting algorithm is listed as Fig. 3.7.

The recursion for the messages in the forward direction is equivalent to the Kalman filter with each forward message $\overrightarrow{\mu}_t$ equivalent to the filtering distribution $\mathbf{P}(x_t \mid z_{1:t})$. The local posteriors after message passing, $\mathcal{B}_t(x_t, x_{t+1})$, are equivalent to the smoothing distribution $\mathbf{P}(x_t, x_{t+1} \mid z_{1:T})$.

procedure ForwardBackward($\{\phi_t\}_{t=1,\dots,T}$)
  $\{\overrightarrow{\mu}_t\} \coloneqq$ ForwardPass($\{\phi_t\}$) ▷ Compute forward and backward messages
  $\{\overleftarrow{\mu}_t\} \coloneqq$ BackwardPass($\{\phi_t\}$)

  $\mathcal{B}_1(x_1) \coloneqq \phi_1(x_1) \cdot \overleftarrow{\mu}_2(x_1)$ ▷ Compute posterior beliefs
  for $t = 2, \dots, T-1$ **do**
    $\mathcal{B}_t(x_{t-1}, x_t) \coloneqq \overrightarrow{\mu}_{t-1}(x_{t-1}) \cdot \phi_t(x_{t-1}, x_t) \cdot \overleftarrow{\mu}_{t+1}(x_t)$ ▷ Eq. 3.83
  end for
  $\mathcal{B}_T(x_{T-1}, x_T) \coloneqq \overrightarrow{\mu}_{T-1}(x_{T-1}) \cdot \phi_T(x_{T-1}, x_T)$
  return $\{\mathcal{B}_t\}$ ▷ Return the set of posterior beliefs
end procedure

procedure ForwardPass($\{\phi_t\}$)
  $\overrightarrow{\mu}_1(x_1) \coloneqq \phi_1(x_1)$
  for $t = 2, \dots, T-1$ **do**
    $\overrightarrow{\mu}_t(x_t) \coloneqq \mathbf{max}_{x_{t-1}} \left[\overrightarrow{\mu}_{t-1}(x_{t-1}) \cdot \phi_t(x_{t-1}, x_t) \right]$ ▷ Eq. 3.81
  end for
  return $\{\overrightarrow{\mu}_t\}$ ▷ Return the computed set of forward messages
end procedure

procedure BackwardPass($\{\phi_t\}$)
  $\overleftarrow{\mu}_T(x_{T-1}) \coloneqq \mathbf{max}_{x_T} \left[\phi_T(x_{T-1}, x_T) \right]$
  for $t = T-1, \dots, 2$ **do**
    $\overleftarrow{\mu}_t(x_{t-1}) \coloneqq \mathbf{max}_{x_t} \left[\phi_t(x_{t-1}, x_t) \cdot \overleftarrow{\mu}_{t+1}(x_t) \right]$ ▷ Eq. 3.82
  end for
  return $\{\overleftarrow{\mu}_t\}$ ▷ Return the computed set of backward messages
end procedure

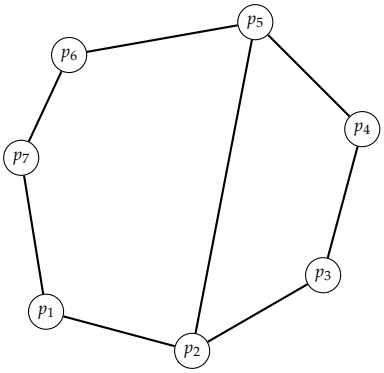
Figure 3.7: The Forward-Backward algorithm for Linear Dynamical Systems

3.7 Gaussian Belief Propagation: General Graphs

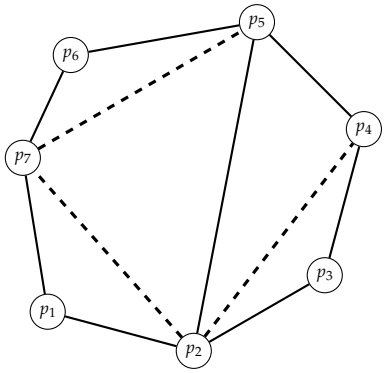
The previous section introduced the intuition for Gaussian Belief Propagation and the practical algorithm for Markov Chains. In this section we will describe how to apply the same techniques to more general problem structures. The generalized procedure requires a substantial amount of preprocessing in order to generate the appropriate graph structures. First, we must convert the factor graph representation of the problem to a Markov Random Field. The Markov Random Field is then used to compute a variable elimination ordering which in turn generates an elimination tree. At the final stage of the preprocessing, the elimination tree is modified to produce a clique tree. This final clique tree data structure

can be used by the Belief Propagation algorithm to partition the sparse global problem into a tree-structured network of small dense local sub-problems. The four stages of this preprocessing are shown in Fig. 3.8. The algorithms presented here are based on Dawid et al. [95] which should be consulted for further details.

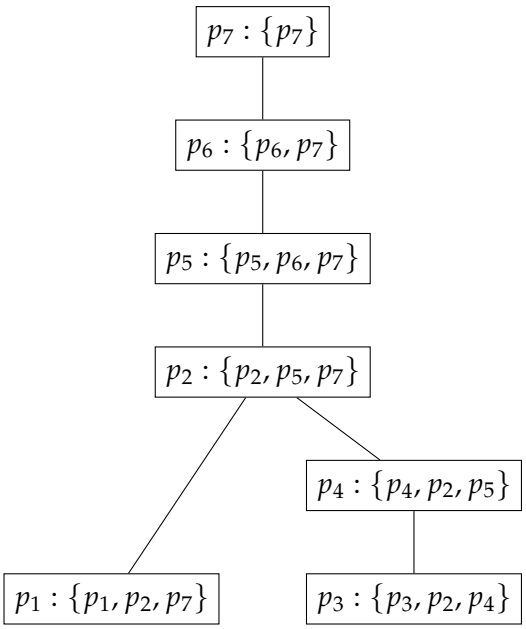
To describe the general case, it is easiest to use Markov Random Fields (MRFs) as a representation of sparsity. A MRF graph contains a vertex for every variable in the problem and an edge between two variables whenever there is a direct relationship between them. Specifically, an edge between two variables indicates that they are present together in the domain of at least one objective/local potential. To convert from a factor graph to an MRF representation, we simply remove all of the factors in the factor graph. For each factor removed, edges are added so that the variables in its domain form a clique. As an example, Fig. 3.8a demonstrate the MRF corresponding to the factor graph in Fig. 3.2.



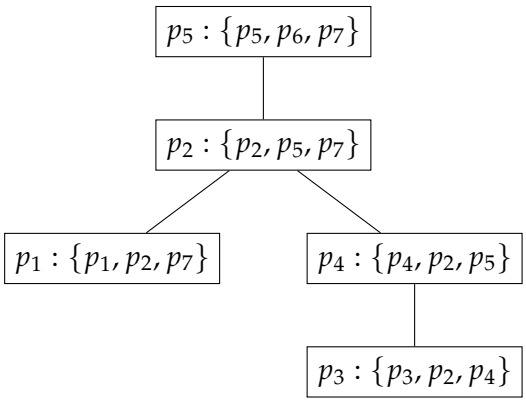
(a) Markov Random Field corresponding to the factor graph in Fig. 3.2.



(b) Triangulation induced by the elimination ordering $p_1, p_3, p_4, p_2, p_5, p_6, p_7$. Dashed edges have been added by the triangulation.



(c) Elimination tree corresponding to the elimination ordering.



(d) Clique tree generated from the elimination tree by keeping only the vertices corresponding to maximal cliques.

Figure 3.8: Illustration of the graphs involved in converting the factor graph in Section 3.2 to a clique tree which can be used for inference or optimization.

3.7.1 Variable Elimination Orderings

The concept of an elimination ordering is central to generalizing Section 3.6. To do so we consider the problem of efficiently finding the maximal likelihood value itself (or marginal likelihood when computing a posterior). This is done by eliminating variables from the problem one at a time via partial maximization (or marginalization). Each elimination solves the optimization problem in terms of a single variable and produces a smaller, equivalent problem. However, each elimination has the potential to offset the decrease in problem size by also decrease sparsity. To avoid this, it is important to choose an elimination order carefully. Once the ordering is established, it can be used to compute the actual value of all variables by using a generalization of the dynamic programming/message passing techniques in Section 3.6.

The process for eliminating a variable starts by collecting all factors which are adjacent to it in the factor graph. This set of factors is replaced by a single combined factor where the target variable has been eliminated in closed form. The domain of the newly created factor is the set of neighbors of the target variable in the MRF. The actual elimination than reduces the size of this domain by one. In the three variable example from Section 3.6.1, the optimal elimination order is x_1, x_2, x_3 (or the reverse). This ordering corresponds to the following expression for computing the maximum likelihood value

$$v^* = \max_{x_3} \Big\{ \underbrace{\max_{x_2} \big\{ \underbrace{\mathbf{P}(x_3 \mid x_2) \max_{x_1} \{ \underbrace{\mathbf{P}(x_1)\, \mathbf{P}(x_2 \mid x_1)}_{\psi_1(x_1, x_2)} \}}_{\psi_2(x_2, x_3)} \big\}}_{\psi_3(x_3)} \Big\} \tag{3.84}$$

Eliminating the variables in the wrong order, for example the ordering x_2, x_3, x_1, corresponds to the expression

$$v^* = \max_{x_1} \Big\{ \underbrace{\max_{x_3} \big\{ \underbrace{\max_{x_2} \{ \underbrace{\mathbf{P}(x_1)\, \mathbf{P}(x_2 \mid x_1)\, \mathbf{P}(x_3 \mid x_2) \}}_{\psi_1(x_1, x_2, x_3)} \}}_{\psi_2(x_1, x_3)} \big\}}_{\psi_3(x_1)} \Big\} \tag{3.85}$$

By eliminating x_2 in the inner-most part of the evaluation, we are forced to incorporate all factors involving x_2 into a single potential and do not gain anything from the problem sparsity. Eliminating blocks of variables in the appropriate order, however, can reduce the total computational effort. Unfortunately finding the optimal elimination order in an arbitrary graph is an NP-complete problem, the solution of which requires heuristics in practice [96].

Heuristic algorithms for variable elimination are based on considering the sequence of temporary potentials (ψ_1, ψ_2, ψ_3) created in the elimination process, as labeled above in Eq. 3.84 and Eq. 3.85. In the case of the the sub-optimal ordering shown in Eq. 3.85, the ψ_1 temporary potential is defined over all three variables, and so this ordering negates any possible savings due to sparsity. In contrast, Eq. 3.84 never constructs a potential with domain size greater than two. To visualize the resulting loss of sparsity, we can modify the original MRF by adding the edges necessary to represent the full sequence of temporary potentials. The resulting graph is the 'triangulation' of the original MRF. The edges added in this way are referred to as fill-in; our goal will be to minimize the amount of such added edges. Note that in the Markov Chain example from the previous section, the natural sequential ordering of the variables is optimal and does not cause any fill-in.

The typical heuristic for computing the elimination ordering, shown in Fig. 3.9, always attempts to eliminate the variable which will minimize fill-in. To pick which of the remaining variables to eliminate at step i, the algorithm attempts to minimize the size of the domain of the temporary potential ϕ_i. As a side effect, we also generate a set of temporary potential domains $\{C[x_i]\}_i$ which correspond to cliques in the triangulated graph. Applying this heuristic to the MRF in Fig. 3.8a results in the triangulated graph shown in Fig. 3.8b where the dashed edges represent fill-in. The corresponding ordering is: $(p_1, p_3, p_4, p_2, p_5, p_6, p_7)$.

```
1: procedure EliminationOrder(𝒢 ≡ (𝒱, ℰ), H)
2:     for i ∈ {1, ..., |𝒱|} do
3:         u := argmin_{v∈𝒢.𝒱} H(𝒢, v)          ▷ Pick next variable using heuristic H
4:         N := neighbors(𝒢, u)
5:         𝒢 := 𝒢\{u}                          ▷ Remove u along with all adjacent edges
6:         ℰ := ℰ ∪ {(i, j)|i, j ∈ N}          ▷ Add induced edges between neighbors
7:         O[u] := i                            ▷ Store vertex ordering
8:         C[u] := N ∪ {u}                      ▷ Store the elimination clique
9:     end for
10:    return (O, C)                            ▷ Return the ordering and associated cliques
11: end procedure

12: procedure H(𝒢, v)
13:    return |neighbors(𝒢, v)|                 ▷ The minimum degree heuristic
14: end procedure
```

Figure 3.9: Computing a variable elimination ordering for a Markov Random Field.

3.7.2 Elimination Trees

The variable elimination ordering computed using Fig. 3.9 is in fact not a total ordering, but a partial order relation. There are sequences of variables which must be eliminated in a fixed order, but different portions of the graph are often independent of each other and can in fact be eliminated in parallel. This gives rise to the concept of an elimination tree. Each node in the elimination tree corresponds to a variable. The root of the tree corresponds to the last variable to be eliminated. The partial ordering is encoded by the rule that a parent can only be eliminated after all of its children, whereas siblings can be eliminated in any order. An example of an elimination tree is shown in Fig. 3.8c. Each node is labeled as $v : \{c_1, c_2, \dots\}$ where v is the variable being eliminated and $\{c_1, c_2, \dots\}$ is the domain of the required temporary potential.

An elimination tree can be build with Fig. 3.10 using the previously computed elimination ordering O and cliques C. For each vertex v in the ordering O, v must be eliminated before the set of neighbors $C[v]$ to ensure that the domains of the temporary potential is not altered. This can be guaranteed by making sure that v is eliminated before any $u \in \text{neighbors}(v)$.

1: **procedure** EliminationTree($\mathcal{V}, O, C$)
2: $\mathcal{E} \coloneqq \emptyset$ ▷ $\mathcal{T} \equiv (\mathcal{V}, \mathcal{E})$
3: **for** $v \in \mathcal{V}$ **do**
4: $N \coloneqq C[v] \backslash \{v\}$ ▷ Clique of non-eliminated neighbors when v is eliminated
5: $p \coloneqq \text{argmin}_{u \in N} \{O[u]\}$ ▷ Parent is next (in O) neighbor to be eliminated
6: $\mathcal{E} \coloneqq \mathcal{E} \cup (p, v)$ ▷ Create a $p \rightarrow v$ edge in $\mathcal{T}$
7: **end for**
8: **return** $\mathcal{T} \equiv (\mathcal{V}, \mathcal{E})$ ▷ Return the elimination tree
9: **end procedure**

Figure 3.10: Constructing an elimination tree from the vertices $\mathcal{V}$, ordering O, and the associated cliques C.

3.7.3 Clique Trees

A clique tree is the final data structure needed in order to perform inference. It is almost identical to the elimination tree, but can be more compact by grouping long chains of eliminations into a single operation. The variable sets in the elimination tree correspond to domains of the temporary potentials, each of which forms a clique in the triangulated graph. Each node in the clique tree corresponds to a *maximal* clique in the triangulated graph. A clique tree is a network of these maximal cliques which encode a whole set of elimination orderings as shown in Fig. 3.8d. To extract a particular elimination ordering, we must choose which set of variables we would like to be eliminated last and define this clique as the root of the tree. The resulting elimination ordering is always consistent with the triangulated graph. The elimination process starts at the leaves of the clique tree and proceeds towards the root. Traversing an edge from node u to v requires eliminating all variables in the set $C[u] \backslash C[v]$. In Fig. 3.8d traversing the path $p_3 \rightarrow p_4 \rightarrow p_2 \rightarrow p_5$ is equivalent to eliminating the variables $\{p_3\}$, $\{p_4\}$, followed by $\{p_2\}$. The path $p_5 \rightarrow p_2 \rightarrow p_1$ corresponds to eliminating $\{p_6, p_7\}$, followed by $\{p_5\}$.

An elimination tree from the previous section can be converted into a clique tree using the algorithm shown in Fig. 3.11. This algorithm is based on two properties of the elimination tree. First, each maximal clique of the triangulated graph has a corresponding node in the elimination tree (the converse is not true). Second, the elimination cliques in the tree are nested. That is, variables are always eliminated as we move up the tree. This im-

plies the maximal cliques can be identified by comparing the elimination set of each node with that of its children. Removing the non-maximal cliques in the elimination tree results in a clique tree which can be used for inference. The removal of non-maximal cliques is performed using a contraction operation on edges in the tree. Whenever a clique is found which is a subset of one of its children, the two vertices are merged by contracting their shared edge. After running this algorithm, all remaining elimination sets are guaranteed to be maximal and so the elimination tree becomes a clique tree. The clique tree for the pose graph example is shown in Fig. 3.8d.

```
1:  procedure CliqueTree(𝒯 ≡ (𝒱, ℰ), O, C)
2:      for v ∈ 𝒱 do                          ▷ Go through all nodes in the elimination tree
3:          for u ∈ children(v) do
4:              if C[v] ⊂ C[u] then           ▷ If v has a child whose clique is a super set
5:                  u* := contract(u, v)      ▷ Merge u and v into a single vertex
6:                  C[u*] := C[u]             ▷ Set the clique of u* to the larger clique
7:                  𝒯 := (𝒯\{u, v}) ∪ {u*}    ▷ Replace u and v with u*
8:                  break
9:              end if
10:         end for
11:     end for
12:     return 𝒯 ≡ (𝒱, ℰ)                     ▷ Return the clique tree
13: end procedure
```

Figure 3.11: Converting an elimination tree into a clique tree.

3.7.4 Belief Propagation on Clique Trees

As already mentioned, a clique tree encodes a set of variable elimination orderings to efficiently solve for the values of any maximal clique in the triangulated graph. The Forward-Backward algorithm described in Section 3.6 used dynamic programming and temporary message potentials to simultaneously compute the posterior over all variable in the chain. The full Belief Propagation algorithm is a generalization of this idea to the clique tree. It simultaneously computes all possible sequences of variable eliminations encoded by the tree and so solves the problem for all variables. For example, the expression for the local

posterior at over the C_2 domain is given by the following formula

$$\mathcal{B}_2(p_2, p_5, p_7) = \phi_2 \cdot \mu_{5,2} \cdot \mu_{1,2} \cdot \mu_{4,2} = \tag{3.86}$$

$$= \phi_2 \cdot \underbrace{\left(\max_{p_6} [\phi_5]\right)}_{\mu_{5,2}} \cdot \underbrace{\left(\max_{p_1} [\phi_1]\right)}_{\mu_{1,2}} \cdot \underbrace{\left(\max_{p_4} [\phi_4 \cdot \mu_{3,4}]\right)}_{\mu_{4,2}} \tag{3.87}$$

$$= \phi_2 \cdot \underbrace{\left(\max_{p_6} [\phi_5]\right)}_{\mu_{5,2}} \cdot \underbrace{\left(\max_{p_1} [\phi_1]\right)}_{\mu_{1,2}} \cdot \underbrace{\left(\max_{p_4} [\phi_4 \cdot \underbrace{\max_{p_3} [\phi_3]}_{\mu_{3,4}}]\right)}_{\mu_{4,2}} \tag{3.88}$$

The general dynamic programming recursion for the posterior with a clique tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is

$$\mathcal{B}_i = \phi_i \cdot \prod_{j \in \text{NEIGHBORS}(i)} \mu_{j,i} \qquad \forall i \in \mathcal{V} \tag{3.89}$$

$$\mu_{i,j} = \max_{C[i] \setminus C[j]} \left[\phi_i \cdot \prod_{k \in \text{NEIGHBORS}(i) \setminus \{j\}} \mu_{k,i} \right] \qquad \forall (i,j) \in \mathcal{E} \tag{3.90}$$

The message potentials $\mu_{i,j}$ are labeled with two indices specifying, respectively, the source and target vertex in the clique tree. This recursion makes it possible to simultaneously compute the posterior over all domains. In practice, the recursive equations are implemented in two rounds of message passing as shown in Fig. 3.13. The initial upward pass computes all the messages from the leaves to the root of the clique tree. The downward pass computes the messages from the root back towards the leaves. Fig. 3.12 illustrates the algorithm on the clique tree example from the previous section.

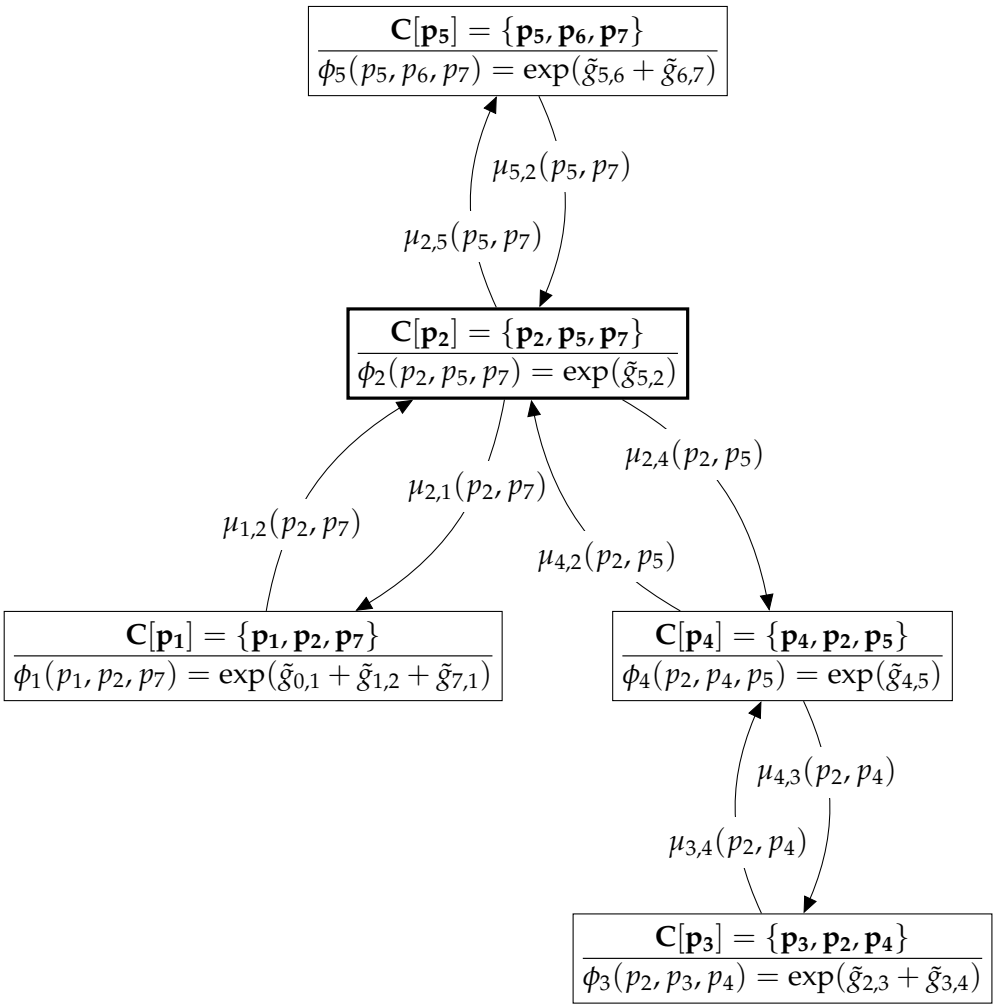


Figure 3.12: Clique tree with annotated messages; the message passing root outlined in bold. Fig. 3.13 lists the general algorithm for computing the messages and posterior beliefs.

procedure BELIEFPROPAGATION($\mathcal{T} \equiv (\mathcal{V}, \mathcal{E}), \{\phi_v\}_{v \in \mathcal{V}}$)
  $R \coloneqq \text{ROOT}(\mathcal{T})$
  UPWARDPASS(R) ▷ Gather information from the leaves upward
  DOWNWARDPASS(R) ▷ Distribute information from the root downward
  for $v \in \mathcal{V}$ **do** ▷ Compute the posterior beliefs
    $\mathcal{B}_v(x_v) \coloneqq \phi_v(x_v) \cdot \prod_{u \in \text{NEIGHBORS}(v)} \mu_{u,v}(x_{u \cap v})$
  end for
  return $\{\mathcal{B}_v\}_{v \in \mathcal{V}}$ ▷ Return the set of posterior beliefs
end procedure

procedure UPWARDPASS(v)
  for $c \in \text{CHILDREN}(v)$ **do**
    UPWARDPASS(c)
  end for
  $p \coloneqq \text{PARENT}(v)$
  $\mu_{v,p}(x_{v \cap p}) \coloneqq \max_{x_{v \setminus p}} \left[\phi_v(x_v) \cdot \prod_{c \in \text{CHILDREN}(v)} \mu_{c,v}(x_{c \cap v}) \right]$
end procedure

procedure DOWNWARDPASS(v)
  for $c \in \text{CHILDREN}(v)$ **do**
    $\mu_{v,c}(x_{v \cap c}) \coloneqq \max_{x_{v \setminus c}} \left[\phi_v(x_v) \cdot \prod_{d \in \text{NEIGHBORS}(v) \setminus \{c\}} \mu_{d,v}(x_{d \cap v}) \right]$
    DOWNWARDPASS(c)
  end for
end procedure

Figure 3.13: The Belief Propagation algorithm over a clique tree. The domain of each vertex/clique is denoted by $x_v \equiv C[v]$. The message domains correspond to intersections of the clique domains and are written as $x_{u \cap v} \equiv C[u] \cap C[v]$. Similarly, $x_{v \setminus u} \equiv C[v] \setminus C[u]$ denotes the set difference domain.

3.8 Discrete Variables and Hybrid Potentials

This dissertation is focused on incorporating discrete variables into the purely continuous problems which can be described with Gaussian Graphical Models and solved with message passing. Up to this point we have reviewed inference and optimization in continuous systems, but have avoided the topic of discrete variables and model selection. This section focuses on adding discrete variables to the Gaussian Potentials from Section 3.6.2 and demonstrates why this addition breaks the inference algorithms discussed above.

As a first step, we will define the notation for local potentials which depend on discrete variables. A hybrid potential $\phi(x, d)$ is a function of both continuous variables $x \in \mathbb{R}^n$ and discrete variables $d \in \{1, \ldots, K\}$. Such a hybrid potential is essentially a table of K different Gaussian Potentials, each conditioned on a possible value of d

$$\phi(x, d{=}k) \equiv \phi^{(k)}(x) \qquad \forall k \in \{1, \ldots, K\} \tag{3.91}$$

$$\phi^{(k)}(x) \equiv \frac{1}{2} x^\top A^{(k)} x + x^\top b^{(k)} + c^{(k)} \qquad \forall k \in \{1, \ldots, K\} \tag{3.92}$$

The parameterization of the discrete potential $\phi(x, d)$ is formally defined as a table of quadratic forms, each one equivalent to one of the Gaussian Potentials in $\{\phi^{(k)}\}$

$$A_\phi^{(k)} = A_{\phi^{(k)}} \qquad \forall k \in \{1, \ldots, K\} \tag{3.93}$$

$$b_\phi^{(k)} = b_{\phi^{(k)}} \qquad \forall k \in \{1, \ldots, K\} \tag{3.94}$$

$$c_\phi^{(k)} = c_{\phi^{(k)}} \qquad \forall k \in \{1, \ldots, K\} \tag{3.95}$$

The difficulty with hybrid potentials is that they are not closed under the partial marginalization and maximization operations required for efficient inference. In the case of the $\phi(x, d)$ as defined above, we can define $\mathbf{marg}_d(\phi)$ as

$$\underset{d}{\mathbf{marg}}\big[\phi(x, d)\big] \equiv \sum_{i=1}^{K} \phi(x, d = i) \tag{3.96}$$

Unfortunately, there is no way to compactly represent the right hand side of Eq. 3.96 without explicitly storing the parameters of each $\phi^{(k)}$ in a table and adding up the K different

values whenever we need to evaluate $\mathbf{marg}_d[\phi](x)$ for a particular value of x (the situation with $\mathbf{max}_d[\phi]$ is identical). Without a closed form representation, taking the product of successive hybrid potentials results in a table with an exponentially increasing number of entries. Unlike the Gaussian case where the brute force approach is at worst cubic, hybrid Belief Propagation is fundamentally intractable without approximations. In the following chapter we will introduce our approximation strategy to avoid this problem.

4

Iterative Local Model Selection

4.1 Introduction

The major limiting factor in applying switching models to perception is a lack of general purpose inference algorithms. In order to be useful for the large scale estimation problems described in the introduction, such an algorithm must satisfy a few basic requirements:

- It must take advantage of sparsity and be able to scale to problems with tens of thousands of variables
- It should not require domain specific tuning parameters or heuristics
- It should work reliably, avoiding unpredictable failure modes

This chapter introduces the Iterative Local Model Selection (ILMS) algorithm as a novel approximate inference technique for SLDS models, and validates the algorithm against these basic requirements above.

The initial development of the algorithm is described in the context of Switching Linear Dynamical Systems (SLDS) rather than the more general Conditional Linear Gausian Networks. Chapter 6 will introduce the necessary modifications for the more general case. SLDS time series models are a powerful generalization of Linear Dynamical Systems which

include a set of discrete variables controlling the behavior of the otherwise linear system. The discrete variables form a Markov process whose value acts as a switch controlling which linear model is in play. In other words, the parameters of the observation and transition model are governed by the discrete variable at each point in time. This structure allows a non-linear process to be described as a mixture of a combinatorial number of linear models: one for each possible sequence of discrete states. SLDS models have been studied in diverse fields for decades with applications including Economics [97], Visual Tracking [98, 99, 100, 101], Speech Recognition [102], and tracking the breathing of Sleep Apnea patients [103].

We first provide background on the general problem of inference in SLDS and describe previous work in the field. Following this, Section 4.4 describes the proposed Iterative Local Model Selection algorithm and provides formal proofs of convergence. In order to validate the proposed approach, we evaluate the algorithm on a set of simulated tracking problems. Comparison against a variety of other approximate inference techniques demonstrates better overall performance relative to computational effort.

4.2 Switching Linear Dynamical Systems

A Switching Linear Dynamical System for a set of times $t \in \{1, \ldots, T\}$ is described by three sequences of variables: the discrete states $\{d_t\}$, the continuous states $\{x_t\}$, and the observations $\{z_t\}$. Each $d_t \in \{1, \ldots, K\}$ is a discrete, unobserved model selection variable which evolves according to a discrete Markov chain. For each value of the model selection variable d_t, a different variant of the continuous motion and observation models is selected; this is the 'switching' aspect of the model. The graphical model capturing these relationships is shown in Fig. 4.1. Assuming the motion and observation models have been

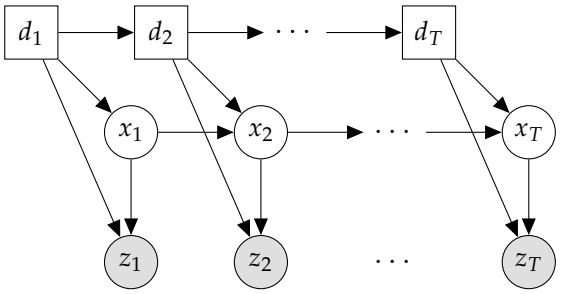


Figure 4.1: The graphical model of a general SLDS.

suitably linearized as $\tilde{f}$ and $\tilde{h}$ respectively, the accompanying transition distributions are

$$x_t \mid x_{i-t}, d_t{=}j \sim \mathcal{N}\left(\tilde{f}^{(j)}(x_{t-1}), \Sigma^{(j)}_{\text{mot}}\right) \tag{4.1}$$

$$z_t \mid x_t, d_t{=}j \sim \mathcal{N}\left(\tilde{h}^{(j)}(x_t), \Sigma^{(j)}_{\text{obs}}\right) \tag{4.2}$$

$$\mathbf{P}(d_t{=}j \mid d_{i-t}{=}i) = T_{ij} \tag{4.3}$$

The algorithms in the following sections will be described in the notation of graphical models, so we define the above model in terms of local potentials. A local potential at time t, denoted by ϕ_t, is the product of the local motion and observation models with the dependency on the observations implicit

$$\phi_t(x_{t-1:t}, d_{t-1}{=}i, d_t{=}j) \equiv \mathbf{P}(x_t \mid x_{t-1}, d_t = j) \cdot \mathbf{P}(z_t \mid x_t, d_t = j) \cdot \mathbf{P}(d_t{=}j \mid d_{t-1}{=}i) \tag{4.4}$$

In this notation the full posterior can be written more compactly as the product of $T-1$ local factors

$$\mathbf{P}(x_{1:t}, d_{1:t} \mid z_{1:t}) \propto \prod_{t=1}^{T-1} \phi_t(x_{t:t+1}, d_{t:t+1}) \tag{4.5}$$

An exact inference algorithm for this problem can be derived by extending Fig. 3.7 with the hybrid local potentials from Eq. 4.4. Modifying the forward and backward message

passing recursion defined in Eq. 3.81 and Eq. 3.82 to include discrete variables results in

$$\begin{aligned}\overrightarrow{\mu}_t(x_t, d_t) &:= \sum_{d_{t-1}} \int \overrightarrow{\mu}_{t-1}(x_{t-1}, d_{t-1}) \phi_t(x_{t-1:t}, d_{t-1:t}) \, \mathrm{d}x_{t-1} \equiv \\ &\equiv \sum_{d_{t-1}} \mathbf{marg}_{x_{t-1}} \left[\overrightarrow{\mu}_{t-1} \cdot \phi_t \right] \end{aligned} \tag{4.6}$$

$$\begin{aligned}\overleftarrow{\mu}_t(x_{t-1}, d_{t-1}) &:= \sum_{d_t} \int \phi_t(x_{t-1:t}, d_{t-1:t}) \overleftarrow{\mu}_{t+1}(x_t, d_t) \, \mathrm{d}x_t \equiv \\ &\equiv \sum_{d_t} \mathbf{marg}_{x_t} \left[\phi_t \cdot \overleftarrow{\mu}_{t+1} \right] \end{aligned} \tag{4.7}$$

These modified equations define a valid belief propagation algorithm, which is unfortunately intractable for any reasonably sized problem. In the purely continuous case $\mathbf{marg}_y \left[\phi(x, y) \right]$ can be represented in closed form via the three parameters of the associated quadratic form. In the case of a hybrid potential $\phi(x, d)$, however, the function $f(x) = \sum_i \phi(x, d{=}i) = \sum_i \phi^{(i)}(x)$ cannot be represented in closed form. An exact representation of f must maintain the full set of parameters for all $\phi^{(i)}$: $\left\{ (A_\phi^{(i)}, b_\phi^{(i)}, c_\phi^{(i)}) \right\}_{i=1}^K$. Representing the product of two such functions will require maintaining K^2 parameters, meaning that each computation of Eq. 4.6 in Fig. 3.7 multiplies the number of components by an additional factor of K. At $t = T$, this will result in a message with K^T different sets of parameters. The combinatorial explosion in the size of the representation makes such a direct approach intractable. This observation is reinforced by the fact that exact inference in hybrid systems is provably NP-hard [104].

4.3 Related Work

An 'exact' hybrid inference method is proposed by Lauritzen [105] and described in detail in Dawid et al. [95]. The authors show that under certain conditions, the exact first and second moments of the posterior can be computed efficiently. Unfortunately a specific form of separability is required between the discrete and continuous variables which does not hold in the SLDS. With exact inference impractical, the majority of previous work has focused on tractable approximations and sampling schemes.

Markov Chain Monte Carlo is naturally well suited for hybrid discrete/continuous

problems. Sampling specific values of the variables allows both discrete and continuous variables to be treated on an equal footing; all that is required is the evaluation of the likelihood at each sample. The downside is that we are forced to pay the performance penalty involved in sampling even for 'simple' unimodal Gaussian variables. This becomes significant in tracking and mapping applications when compared with the traditional continuous optimization and filtering techniques. Oh et al. [101] propose a collapsed/Rao-Blackwellized sampling strategy with a data driven proposal distribution to overcome the slow convergence of vanilla MCMC when applied to the SLDS model. Despite being a viable alternative, we avoid direct evaluations again MCMC algorithms for several reasons. Theoretically, MCMC will always converge to the exact posterior when given enough time, so the performance will naturally depend on how long one is will to wait. On top of this, the choice of proposal distribution has a strong effect on performance. This makes it hard to compare to MCMC without a set application in mind. Oh et al. [101] in particular use a proposal which is essentially learned from observation data and so is also dependent on the choice of learning algorithm. These factors make it difficult to perform a fair comparison in the context of general purpose algorithms.

A notable exception is Nonparametric Belief Propagation (NBP)[106] which generalizes the concept of a particle filter to arbitrary graphical models. The NBP algorithm approximates messages as Mixtures-of-Gaussians and uses MCMC to re-sample new mixtures which represent the various products involved in Belief Propagation. This approach is extremely general, but requires sampling from a product of (possibly high dimensional) mixtures in order to compute each message. Due to this sampling step, as well as the non-parametric representation of each message, NBP requires significantly more computation per message when compared to our approach.

That said, this chapter will focus on comparisons with the popular [99, 107] deterministic algorithms (i.e. Approximate Viterbi [99, 107], GPB2 Smoothing [97, 108], Expectation Propagation [109], and Variational Bayes [103]). With the exception of Variational Bayes, these approximations rely on 'collapsing' a Mixtures-of-Gaussians into a single unimodal

Gaussian distribution which is in some sense close to the original mixture. Approximations of this class are closely related to the Expectation Propagation, Assumed Density Filtering [110], and the Boyen-Koller algorithm [111].

4.3.1 Approximate Viterbi

The Approximate Viterbi[99] algorithm is a two-stage procedure consisting of a filtering and smoothing stage. This technique is related to the Iterative Local Model Selection algorithm presented in this chapter. In the first, filtering stage an estimate of the switching sequence $d^*_{1:T}$ is computed in a single forward pass. Following this, a Kalman Smoother is run on the continuous system with the discrete switching states fixed.

A maximal likelihood approach is taken to approximate the filtered sequence of discrete states. Translated into the message passing notation, the first phase of the algorithm can be thought of as a forward pass of the Belief Propagation algorithm. The exact forward message shown in Eq. 4.6 is approximated by picking the best value of d_{t-1} for each possible value of d_t. The sum over d_{t-1} which is part of the exact marginal is replaced by a single component for each value of d_t=j

$$d_{t-1}|_{d_t=j} := \operatorname*{argmax}_i \left\{ \max_{(x_{t-1},x_t)} \left[\overrightarrow{\mu}_{t-1} \cdot \phi_t \right]^{(d_{t-1}=i,d_t=j)} \right\} \tag{4.8}$$

$$\overrightarrow{\mu}_t(x_t, d_t{=}j) := \operatorname*{marg}_{x_{t-1}} \left[\overrightarrow{\mu}_{t-1} \cdot \phi_t \right]^{\left(d_{t-1}=d_{t-1}|_{d_t=j}, d_t=j\right)} \tag{4.9}$$

where $d_{t-1}|_{d_t=j}$ represents a table of optimal values for d_{t-1} conditioned on d_t=j. The notation $[\cdot]^{(d_t=\cdot,d_{t-1}=\cdot)}$ selects a single Gaussian potential from a hybrid potential as described in Section 3.8. The algorithm computes a table of Gaussian posteriors for each of the K^2 possible values of (d_{t-1}, d_t) using the approximate message $\overrightarrow{\mu}_{t-1}$ as input. Instead of marginalizing the d_{t-1} dimension of the table via summation, however, we pick a single value of d_{t-1} for each d_t. After the forward pass is complete, the full sequence $d^*_{1:T}$ can be reconstructed using the values of $d_{t-1}|_{d_t=j}$. Once $d^*_{1:T}$ is fixed, a regular Kalman Smoother is run to estimate the continuous variables.

4.3.2 GPB2

The GPB2 (Generalized Pseudo-Bayesian Order 2) [97, 108] algorithm is a filtering approach which approximates marginalization of the discrete variables using moment matching. Instead of computing the true marginal, the forward messages are approximated with a single Gaussian potential which has the same mean and covariance as the mixture. To derive the algorithm, we first introduce a table of Gaussian potentials representing an intermediate result of Eq. 4.6. Each of the K^2 entries in the table corresponds to the filtering distribution over x_t conditioned on a value of (d_{t-1}, d_t) with x_{t-1} marginalized out. The two indices of the table ν_{ij} correspond to $d_{t-1} = i, d_t = j$

$$\begin{aligned} \nu_{ij}(x_t) &\equiv \left[\underset{x_{t-1}}{\mathbf{marg}} \left[\overrightarrow{\mu}_{t-1} \cdot \phi_t \right] \right]^{(i,j)} \\ &= \int \overrightarrow{\mu}_{t-1}(x_{t-1}, d_{t-1}{=}i) \phi_t(x_{t-1}, x_t, d_{t-1}{=}i, d_t{=}j) \, \mathrm{d}x_{t-1} \end{aligned} \tag{4.10}$$

The exact forward message can be written as a mixture, $\mathcal{M}_j(x)$, of the Gaussian components of ν_{ij}

$$\overrightarrow{\mu}_t(x_t, d_t{=}j) \equiv \sum_{d_{t-1}=1}^{K} \underset{x_{t-1}}{\mathbf{marg}} \left[\overrightarrow{\mu}_{t-1} \cdot \phi_t \right] \equiv \sum_{i=1}^{K} \nu_{ij}(x_t) \equiv \mathcal{M}_j(x_t) \tag{4.11}$$

GPB2 approximates the forward message using the first and second moments of $\mathcal{M}_j(x)$

$$\overrightarrow{\mu}_t(x_t, d_t{=}j) \approx \mathcal{N}\Big(x_t \,;\, \mathbf{E}_{\mathbf{x}\sim\mathcal{M}_\mathbf{j}}[x], \mathbf{Cov}_{\mathbf{x}\sim\mathcal{M}_\mathbf{j}}[x] \Big) \tag{4.12}$$

While the name GPB2 often refers to the above filtering algorithm [112], a GPB2 smoother was developed in Kim [97]. We will refer to the smoothing version of the algorithm as *GPB2S*. GPB2S is more intricate, but fundamentally relies on decoupling the discrete and continuous variables during the backward pass. As will be demonstrated in the evaluations, this approximation is not always appropriate and can sometimes produce wildly inaccurate results.

4.3.3 Expectation Propagation

A more recent smoothing approach based on Expectation Propagation (EP) [109, 110] also uses moment matching. Unlike GPB2, Expectation Propagation matches the moments of

the posterior rather than filtering distribution. This results in a smoothing algorithm which performs very well, but does not always converge. In practice, the message passing must be damped in order to ensure both convergence and well defined posterior potentials.

As in GPB2, we define a table ν_{ij} to simplify the description of the algorithm. In this case $\nu_{ij}(x_t)$ represents the local posterior conditioned on $d_{t-1} = i, d_t = j$ with x_{t-1} marginalized out (note that the approximate backward message $\overleftarrow{\mu}_{t+1}$ is also included)

$$\begin{aligned} \nu_{ij}(x_t) &\equiv \left[\underset{x_{t-1}}{\mathbf{marg}} \left[\overrightarrow{\mu}_{t-1} \cdot \phi_t \cdot \overleftarrow{\mu}_{t+1} \right] \right]^{(i,j)} \\ &= \int \overrightarrow{\mu}_{t-1}(x_{t-1}, d_{t-1}{=}i) \cdot \phi_t(x_{t-1}, x_t, d_{t-1}{=}i, d_t{=}j) \cdot \overleftarrow{\mu}_{t+1}(x_t, d_t{=}j) \, \mathrm{d}x_{t-1} \end{aligned} \tag{4.13}$$

The exact forward message can be computed from ν_{ij} by summing over i and dividing by the backward message $\overleftarrow{\mu}_{t+1}$

$$\overrightarrow{\mu}(x_t, d_t{=}j) = \frac{\sum_i \nu_{ij}(x_t)}{\overleftarrow{\mu}_{t+1}(x_t, d_t{=}j)} \tag{4.14}$$

Expectation Propagation first uses moment matching to approximate the posterior mixture $\mathcal{M}_j = \sum_i \nu_{ij}$, and then divides by $\overleftarrow{\mu}_{t+1}$ to compute the approximate forward message.

$$\overrightarrow{\mu}(x_t, d_t{=}j) \approx \frac{\mathcal{N}\left(x_t \,;\, \mathbf{E}_{\mathbf{x}\sim\mathcal{M}_\mathrm{j}}[x], \mathbf{Cov}_{\mathbf{x}\sim\mathcal{M}_\mathrm{j}}[x]\right)}{\overleftarrow{\mu}_{t+1}(x_t, d_t{=}j)} \tag{4.15}$$

By approximating the posterior moments, the approximation can incorporate information from both the past and the future. The backward pass of the algorithm uses the same approximation, but divides by the incoming forward message to compute the approximate backward message.

The downside is that the division in Eq. 4.15 can result in an invalid Gaussian potential with a negative information matrix. Even when the potentials are forced to be positive definite, the algorithm still provides no guarantees and will occasionally oscillate or fail due to numerical stability. To help avoid invalid potentials and non-convergence issues the EP algorithm is typically damped. At each computation of a message $\overrightarrow{\mu}_t$, the parameters

of the potentials are averaged with those from the previous version of that message $\overrightarrow{\mu}_t^{\text{last}}$

$$A := \alpha A^{\text{new}} + (1-\alpha) A^{\text{old}} \tag{4.16}$$

$$b := \alpha b^{\text{new}} + (1-\alpha) b^{\text{old}} \tag{4.17}$$

$$c := \alpha c^{\text{new}} + (1-\alpha) c^{\text{old}} \tag{4.18}$$

This has the same effect as a smaller step size in an optimization algorithm and generally improves convergence at the expense of more iterations being required.

4.3.4 Variational Bayes

The variational approach [113, 103, 114, 115] is a generalization of Expectation Maximization [116]. Unlike Expectation Propagation, Variational Bayesian inference provides convergence guarantees. The algorithm is based on approximating the intractable posterior $\mathbf{P}(x_{1:T}, d_{1:T} \mid z_{1:T})$ as a product of two simpler distributions

$$\mathbf{Q}(x_{1:T}, d_{1:T} \mid z_{1:T}) = \mathbf{Q}_{\mathbf{x}}(x_{1:T} \mid z_{1:T})\, \mathbf{Q}_{\mathbf{d}}(d_{1:T} \mid z_{1:T}) \tag{4.19}$$

This approximation decouples inference in the discrete and continuous portions of the model. The graphical models of the approximating distributions $\mathbf{Q}_{\mathbf{x}}$ and $\mathbf{Q}_{\mathbf{d}}$ are shown in Fig. 4.2. Inference in both of these distributions is tractable since each is just a Markov model with a respectively continuous and discrete state space.

(a) $\mathbf{Q}_{\mathbf{d}}(d_1, \ldots, d_T)$

(b) $\mathbf{Q}_{\mathbf{x}}(x_1, \ldots, x_T)$

Figure 4.2: Markov networks representing the structure of the approximate distribution $\mathbf{Q}$.

The variational approximation makes use of our ability to perform efficient inference in the approximate distributions in order to iteratively improve the approximations. At convergence, the approximate distributions are optimal in the sense that they minimize KL divergence from the true posterior

$$\mathrm{D}_{\mathrm{KL}}(\mathbf{P}(x_{1:T}, d_{1:T} \mid z_{1:T}) \| \mathbf{Q_x} \cdot \mathbf{Q_d}) \tag{4.20}$$

This fitting is performed via coordinate ascent over the two approximate distributions. The following two updates are applied until the approximation converges

$$\mathbf{Q}_\mathbf{x}^{(\mathbf{i+1})} := \underset{\mathbf{Q_x}}{\operatorname{argmin}} \left\{ \mathrm{D}_{\mathrm{KL}} \left(\mathbf{P}(x_{1:T}, d_{1:T} \mid z_{1:T}) \middle\| \mathbf{Q_x} \cdot \mathbf{Q}_\mathbf{d}^{(\mathbf{i})} \right) \right\} \tag{4.21}$$

$$\mathbf{Q}_\mathbf{d}^{(\mathbf{i+1})} := \underset{\mathbf{Q_d}}{\operatorname{argmin}} \left\{ \mathrm{D}_{\mathrm{KL}} \left(\mathbf{P}(x_{1:T}, d_{1:T} \mid z_{1:T}) \middle\| \mathbf{Q}_\mathbf{x}^{(\mathbf{i+1})} \cdot \mathbf{Q_d} \right) \right\} \tag{4.22}$$

As demonstrated in Beal [117], applying variational calculus to the above results in the follow general update equations

$$\log \mathbf{Q}_\mathbf{x}^{(\mathbf{i+1})} := \mathbf{E}_{\mathbf{Q}_\mathbf{d}^{(\mathbf{i})}} [\log \mathbf{P}(x_{1:T}, d_{1:T} \mid z_{1:T})] - \log Z_x \tag{4.23}$$

$$\log \mathbf{Q}_\mathbf{d}^{(\mathbf{i+1})} := \mathbf{E}_{\mathbf{Q}_\mathbf{x}^{(\mathbf{i+1})}} [\log \mathbf{P}(x_{1:T}, d_{1:T} \mid z_{1:T})] - \log Z_d \tag{4.24}$$

where Z_x and Z_d are normalization constants. In the update of $\mathbf{Q_x}$ we compute the expected log-likelihood with respect to $\mathbf{Q_d}$. Computing this expectation is equivalent to averaging the different continuous models in log-space and results in a tractable continuous-only Markov chain. Analogously, the update for $\mathbf{Q_d}$ integrates over the continuous variables and results in an averaged discrete model. These updates can be simplified to take advantage of problem structure. Using the linearity of the expectation operator and Eq. 4.5, we get the following simplifications

$$\begin{aligned} \log \mathbf{Q}_\mathbf{x}^{(\mathbf{i+1})} &:= \mathbf{E}_{\mathbf{Q}_\mathbf{d}^{(\mathbf{i})}} \left[\sum_t \log \phi_t(x_{t-1}, x_t, d_{t-1}, d_t) \right] - \log Z_x = \\ &= \sum_t \mathbf{E}_{\mathbf{Q}_\mathbf{d}^{(\mathbf{i})}} [\log \phi_t(x_{t-1}, x_t, d_{t-1}, d_t)] - \log Z_x \end{aligned} \tag{4.25}$$

$$\begin{aligned} \log \mathbf{Q}_\mathbf{d}^{(\mathbf{i+1})} &:= \mathbf{E}_{\mathbf{Q}_\mathbf{x}^{(\mathbf{i+1})}} \left[\sum_t \log \phi_t(x_{t-1}, x_t, d_{t-1}, d_t) \right] - \log Z_d = \\ &= \sum_t \mathbf{E}_{\mathbf{Q}_\mathbf{x}^{(\mathbf{i+1})}} [\log \phi_t(x_{t-1}, x_t, d_{t-1}, d_t)] - \log Z_x \end{aligned} \tag{4.26}$$

In other words, the expectations only have to be computed locally over each potential $\phi_t(x_{t-1}, x_t, d_{t-1}, d_t)$. We can define the expected local log-potentials for the continuous and discrete approximate distributions, respectively, as

$$\log \bar{\phi}_t^{\text{cont}}(x_{t-1}, x_t) \equiv \mathbf{E}_{(\mathbf{d_{t-1}}, \mathbf{d_t}) \sim \mathbf{Q}_\mathbf{d}^{(\mathbf{i})}} [\log \phi_t(x_{t-1}, x_t, d_{t-1}, d_t)] \tag{4.27}$$

$$\log \bar{\phi}_t^{\text{disc}}(d_{t-1}, d_t) \equiv \mathbf{E}_{(\mathbf{x_{t-1}}, \mathbf{x_t}) \sim \mathbf{Q}_\mathbf{c}^{(\mathbf{i+1})}} [\log \phi_t(x_{t-1}, x_t, d_{t-1}, d_t)] \tag{4.28}$$

Note that these are exactly the expectations which can be computed efficiently with belief propagation. By applying belief propagation to the chain of potentials $\left\{\bar{\phi}_t^{\text{disc}}(d_{t-1}, d_t)\right\}_t$, we can efficiently calculate the marginals $\mathbf{Q}_\mathbf{d}^\mathbf{i}(d_{t-1}, d_t)$. These marginals allows us to compute the expected continuous potentials $\left\{\bar{\phi}_t^{\text{cont}}(x_{t-1}, x_t)\right\}_t$. Applying the Gaussian version of Belief Propagation and computing local expectations results in an updated set of expected discrete potentials. Iterating these four steps (continuous belief propagation, compute expected discrete potentials, discrete belief propagation, compute expected continuous potentials) is the essence of the variational inference algorithms described by Ghahramani and Hinton [103], Pavlovic et al. [114], and Oh et al. [115].

The only remaining question is how to compute the local expectations described in Eq. 4.27 and Eq. 4.28. To compute these, consider the definition of the potential ϕ_t in log space

$$\log \phi_t(x_{t-1}, x_t, d_{t-1}{=}i, d_t{=}j) = \frac{1}{2}(x_{t-1}, x_t)^\top A^{(i,j)}(x_{t-1}, x_t) + (x_{t-1}, x_t)^\top b^{(i,j)} + c^{(i,j)} \tag{4.29}$$

Computing the expectation of $\log \phi_t$ over the discrete variables amounts to averaging the parameters of the quadratic forms

$$A_{\bar{\phi}_t^{\text{cont}}} = \sum_{i,j} \mathcal{B}_t(d_{t-1}{=}i, d_t{=}j) \cdot A^{(i,j)} \tag{4.30}$$

$$b_{\bar{\phi}_t^{\text{cont}}} = \sum_{i,j} \mathcal{B}_t(d_{t-1}{=}i, d_t{=}j) \cdot b^{(i,j)} \tag{4.31}$$

$$c_{\bar{\phi}_t^{\text{cont}}} = \sum_{i,j} \mathcal{B}_t(d_{t-1}{=}i, d_t{=}j) \cdot c^{(i,j)} \tag{4.32}$$

where $\mathcal{B}$ represents the Belief Propagation posteriors. For the average discrete potentials,

the computation is a bit more complicated; in this case $\mathcal{B}_t(x_{t-1:t})$ is Gaussian, so

$$\log \bar{\phi}_t^{\text{disc}}(d_{t-1}{=}i, d_t{=}j) = \mathbf{E}_{\mathcal{B}_t}\left[\frac{1}{2}x_{t-1:t}^\top A^{(i,j)} x_{t-1:t} + x_{t-1:t}^\top b^{(i,j)} + c^{(i,j)}\right] \tag{4.33}$$

The expectation of a quadratic form when $x \sim \mathcal{N}(\mu, \Sigma)$ is given by

$$\begin{aligned}\mathbf{E}\left[\frac{1}{2}x^\top Ax + x^\top b + c\right] =& \frac{1}{2}\mathbf{E}\left[x^\top Ax\right] + \mu^\top b + c = \\ =& \frac{1}{2}\operatorname{tr}(A\Sigma) + \frac{1}{2}\mu^\top A\mu + \mu^\top b + c\end{aligned} \tag{4.34}$$

The downside of Variational Bayes is a general tendency to get stuck in local maxima. This is addressed by Ghahramani and Hinton [103] via deterministic annealing. The expected log-potentials are scaled by an annealing term $\alpha \in [0, 1]$. When $\alpha = 0$, the log-potential is uniform. Setting $\alpha = 1$ results in the original update equation. Progressively increasing α from 0 to 1 as the algorithm progresses helps avoid local maxima.

4.4 Iterative Local Model Selection

The algorithm proposed in this thesis, *Iterative Local Model Selection* (ILMS), is most closely related to the *Approximate Viterbi* algorithm described in Section 4.3.2. Instead of computing the sequence of switching states once, however, ILMS iteratively scans the sequence of variables and updates the current estimate of each discrete variable. The current most likely value of each discrete variable d_t is used to compute the forward and backward messages. The algorithm interleaves the computation of d_t^* with smoothing of the continuous variables. Unlike the Approximate Viterbi algorithm and GPB2, ILMS is a smoothing algorithm which takes advantage of all data in the past and future. Compared to EP, ILMS guarantees convergence, does not require damping, and is faster in practice.

4.4.1 Simplified Case

At first we will assume that there is no direct dependency between the discrete variables as illustrated in Fig. 4.3. In the simplified case, the local potential at each t no longer contains

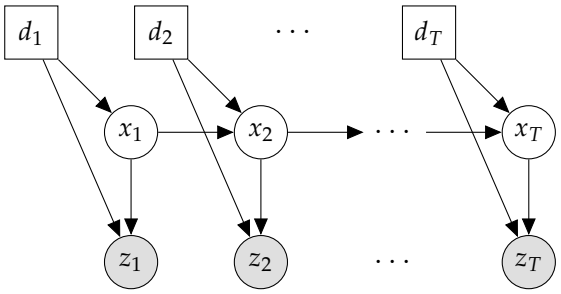


Figure 4.3: The graphical model of the simplified SLDS with $d_t \perp\!\!\!\perp d_{t+1} \mid x_t$.

d_{t-1} in its domain

$$\phi_t(x_{t-1}, x_t, d_t) = \mathbf{P}(z_t \mid x_t, d_t) \cdot \mathbf{P}(x_t \mid x_{t-1}, d_t) \cdot \mathbf{P}(d_t) \tag{4.35}$$

Later, the algorithm will be generalized to handle the full SLDS model.

To introduce the algorithm, we will at first assume that the set of discrete model selection variables is fully specified, with $d_t = d_t^*$. In this case Belief Propagation can be used efficiently as in Fig. 3.7. After a run of BP, we will have a set of forward messages $\{\overrightarrow{\mu}_t\}$ and backward messages $\{\overleftarrow{\mu}_t\}$ computed under the assumption that $d_{1:T} = d_{1:T}^*$.

$$\overrightarrow{\mu}_t(x_t) = \underset{x_{t-1}}{\mathbf{marg}} \left[\phi_t^{(d_t^*)} \cdot \overrightarrow{\mu}_{t-1}\right] \tag{4.36}$$

$$\overleftarrow{\mu}_t(x_{t-1}) = \underset{x_t}{\mathbf{marg}} \left[\phi_t^{(d_t^*)} \cdot \overleftarrow{\mu}_{t+1}\right] \tag{4.37}$$

Note that the notation $\phi_t^{(d_t^*)} \equiv \phi_t(x_{t-1}, x_t, d_t{=}d_t^*)$ refers to a single purely Gaussian element in the hybrid potential ϕ_t. Using ϕ_t combined with $\overrightarrow{\mu}_{t-1}$ and $\overleftarrow{\mu}_{t+1}$, we can compute the posterior at t

$$\mathcal{B}_t = \overrightarrow{\mu}_{t-1} \cdot \phi_t^{(d_t^*)} \cdot \overleftarrow{\mu}_{t+1} \tag{4.38}$$

In this expression for the posterior, the incoming messages do not depend on the value of d_t. The dependence of $\mathcal{B}_t$ on the chosen value of d_t is only through the local potential ϕ_t

$$\mathcal{B}_t(x_{t-1}, x_t) = \underbrace{\overrightarrow{\mu}_{t-1}(x_{t-1}) \cdot \overleftarrow{\mu}_{t+1}(x_t)}_{\text{implicit dependence on } (d_{1:t-1}, d_{t+1:T})} \cdot \underbrace{\phi_t(x_{t-1}, x_t, d_t{=}d_t^*)}_{\text{dependence on } d_t} \tag{4.39}$$

Consider what would happen if we assigned a different value to d_t. If we set $d_t = j$, the local posterior can be computed using the same incoming message potentials by multiplying in

a different versions of the local potential. This results in a different local posterior, $\mathcal{B}^{(j)}$, for each j

$$\mathcal{B}_t^{(j)} = \overrightarrow{\mu}_{t-1} \cdot \phi_t^{(j)} \cdot \overleftarrow{\mu}_{t+1} \tag{4.40}$$

We can use this property to pick a new value for d_t, while holding the other discrete variables fixed. Once d_t is updated, any message which was computed based on its previous value is invalidated. It can be seen from Eq. 4.36 and Eq. 4.36 that the outgoing messages at t, $\overleftarrow{\mu}_t$ and $\overrightarrow{\mu_t}$, will be invalidated by changing the value of d_t. Any messages which are computed using these two as input will also be invalidated, so changing d_t will invalidate the set of forward messages after time t and backward messages prior to time t: $\{\overleftarrow{\mu}_{t'} | t' \leq t\} \cup \{\overrightarrow{\mu}_{t'} | t' \geq t\}$.

ILMS operates by interleaving the computation of messages and updates to the discrete variables. After updating d_t, the algorithm recomputes $\overrightarrow{\mu}_t$ so that both $\overrightarrow{\mu}_t$ and $\overleftarrow{\mu}_{t+2}$ are valid. This allows $\mathcal{B}_{t+1} = \overrightarrow{\mu}_t \cdot \phi_{t+1} \cdot \overleftarrow{\mu}_{t+2}$ to be computed and d_{t+1} to be updated in turn. When we reach the end of the sequence, the algorithm is reversed and the same procedure is applied while recomputing the backward messages. This forward and backward scanning is repeated until the values of the discrete variables stop changing and the algorithm converges.

The criterion by which new values of d_t are picked is not specified by the above description of the algorithm. There are two choices examined in this thesis: maximal likelihood (ML) and maximal marginal likelihood (MM). The maximal likelihood version of the algorithm attempts to find a discrete and continuous state which maximizes the overall likelihood

$$(d_{1:T}^*, x_{1:T}^*) = \underset{(d_{1:T}, x_{1:T})}{\operatorname{argmax}} \left\{ \mathbf{P}(x_{1:T}, d_{1:T} \mid z_{1:T}) \right\} \tag{4.41}$$

This naturally corresponds to MAP inference and max-marginal message passing (i.e. using $\mathbf{max}[\cdot]$ as the marginalization operator).

The second version attempts to maximizes the marginal likelihood of the discrete variables with the continuous variables integrated out. This corresponds to a Bayesian model

selection approach where we first pick a value of $d_{1:T}$, and then a value for $x_{1:T}$ conditioned on it

$$d^*_{1:T} = \operatorname*{argmax}_{d_{1:T}} \left\{ \mathbf{P}(d_{1:T} \mid z_{1:T}) \right\} = \operatorname*{argmax}_{d_{1:T}} \left\{ \int \mathbf{P}(x_{1:T}, d_{1:T} \mid z_{1:T}) \, \mathrm{d}x_{1:T} \right\} \tag{4.42}$$

$$x^*_{1:T} = \operatorname*{argmax}_{x_{1:T}} \left\{ \mathbf{P}(x_{1:T} \mid d^*_{1:T}, z_{1:T}) \right\} \tag{4.43}$$

This version of the algorithm corresponds to using $\mathbf{marg}[\cdot]$ in the message passing.

When we are updating a single d_t, both of these quantities can be computed and maximized using only the local posterior $\mathcal{B}_t$. For the ML version, d_t is updated using

$$\begin{aligned} d^*_t &:= \operatorname*{argmax}_j \left\{ \max_{x_{1:T}} \left\{ \mathbf{P}(x_{1:T}, d_{1:t-1}{=}d^*_{1:t-1}, d_t{=}j, d_{t+1:T}{=}d^*_{t+1:T} \mid z_{1:T}) \right\} \right\} = \\ &= \operatorname*{argmax}_j \left\{ \max_{x_{t-1:t}} \left\{ \mathcal{B}^{(j)}(x_{t-1:t}) \, \mathrm{d}x_{t-1:t} \right\} \right\} \end{aligned} \tag{4.44}$$

In the case of MM, the update equation is

$$\begin{aligned} d^*_t &:= \operatorname*{argmax}_j \left\{ \mathbf{P}(d_{1:t-1}{=}d^*_{1:t-1}, d_t{=}j, d_{t+1:T}{=}d^*_{t+1:T} \mid z_{1:T}) \right\} = \\ &= \operatorname*{argmax}_j \left\{ \int \mathcal{B}^{(j)}(x_{t-1:t}) \, \mathrm{d}x_{t-1:t} \right\} \end{aligned} \tag{4.45}$$

The description of the ILMS algorithm does not depend on which criterion is used for this update, so we will ignore this distinction until Section 4.5 where we compare the two versions. Fig. 4.4 illustrates the process of computing an outgoing message using Eq. 4.45.

Both Eq. 4.45 and Eq. 4.44 are optimizing over a single discrete d_t together with *all* continuous $x_{1:T}$. If we perform this optimization for each t, we are performing block coordinate ascent over the overlapping blocks $\{(d_t, x_{1:T})\}_t$, with the remaining discrete variables, $(d_{1:t-1}, d_{t+1:T})$, held fixed. The ILMS algorithm for the simplified SLDS problem is shown in Fig. 4.5.

ILMS approximates the mixture model of each forward and backward message with a single component selected based on the value of the local posterior just prior to the message being computed. In this sense it is similar to Expectation Propagation, except that the mixture distributions are approximated using a single mode rather then their

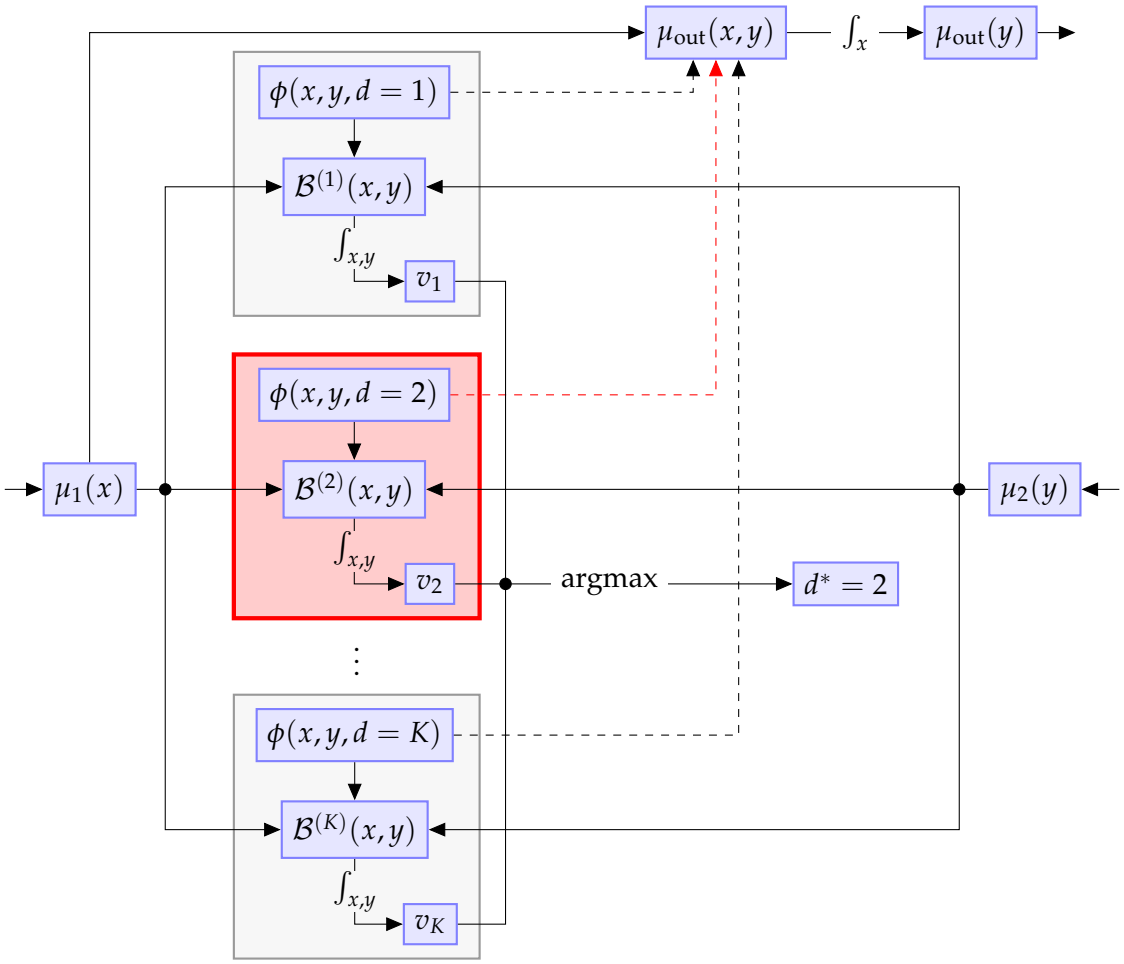


Figure 4.4: A diagram of ILMS local maximization procedure. Each arrow indicates its source being multiplied into its destination, unless specifically labeled otherwise. The incoming potentials from this clique's neighbors (μ_1 and μ_2) are multiplied with each discrete mode of the local potential ϕ to create K different candidate posteriors: $\mathcal{B}^{(1)},\ldots,\mathcal{B}^{(K)}$. For each candidate, $v_k = \int \mathcal{B}^{(k)}$ computes the marginal likelihood conditioned on $d=k$. In this illustration, selecting $d^* = \text{argmax}_k \{v_k\} = 2$ maximizes the marginal likelihood. The outgoing message μ_{out} is then computed assuming $d=2$. The dashed arrows represent the set of operations which are controlled by the value of d^*. Out of these, only the red dashed arrow corresponding to $\phi(x,y,d{=}2)$ is active since $d^*=2$ was selected.

moments. While this approximation is intuitively more greedy, it comes with the advantage of guaranteed convergence.

Proposition 1. The simplified ILMS procedure shown in Fig. 4.5 converges.

Proof. Consider the sequence of optimizations being performed in Eq. 4.45 during a forward pass of the algorithm after at least one backward pass has been completed (to ensure all forward and backward messages have been computed at least once). At time t, updating the value of d_t^* results in new values for $\overrightarrow{\mu}_t$ and $\overleftarrow{\mu}_t$.

Since the messages $\{\overrightarrow{\mu}_{t'}\}_{t'>t}$ and $\{\overleftarrow{\mu}_{t}\}_{t'<t}$ implicitly depend on the value of d_t^*, the cached versions of these messages become invalid after the update. However, all other cached messages remain valid and $\overrightarrow{\mu}_t$ is immediately recalculated. This means that after updating $\mathcal{B}_t$, d_t^*, $\overrightarrow{\mu}_t$, and $\mathcal{B}_{t+1}$ but before updating d_{t+1}^* we are in a state where both $\mathcal{B}_t$ and $\mathcal{B}_{t+1}$ are a valid representations of the local posterior.

Let $v_t(k) = v_k$ as computed in line 31 of Fig. 4.5. In this case, $v_t(d_t^*) = v_{t+1}(d_{t+1}^*)$ since both values are computed based on a valid representation of the posterior. Updating d_{t+1}^* cannot decrease the value of the expression $v_{t+1}(d_{t+1}^*)$, so the forward-pass cannot decrease the likelihood. By analogy, subsequent backward passes cannot decrease the likelihood and the algorithm as a whole must converge because the likelihood of the SLDS is bounded. □

procedure ILMS_SIMPLIFIED($\{\phi_t\}_{t=1,\dots,T}$)
  for $t = 1, \dots, T$ **do** ▷ Initialize messages
    $\overrightarrow{\mu}_t(x_t) := \mathbf{1}$
    $\overleftarrow{\mu}_t(x_{t-1}) := \mathbf{1}$
  end for
  while not converged **do**
    FORWARDPASS()
    BACKWARDPASS()
  end while
  return $\left(\{\mathcal{B}_t\}, \{d_t^*\}\right)$ ▷ Return local beliefs and discrete states
end procedure

procedure FORWARDPASS()
  for $t = 1, \dots, T$ **do**
    $(\mathcal{B}_t, v) := \text{ILMS\_UPDATE}(t)$
    $v_t^* := \max_k \{v_k\}$
    $d_t^* := \operatorname{argmax}_k \{v_k\}$ ▷ Update local discrete state
    $\overrightarrow{\mu}_t(x_t) := \int \overrightarrow{\mu}_{t-1}(x_{t-1})\phi_t(x_{t-1:t}, d_t{=}d_t^*)\,\mathrm{d}x_{t-1} \equiv \mathbf{marg}_{x_{t-1}}\left[\overrightarrow{\mu}_{t-1} \cdot \phi_t^{(d_t^*)}\right]$
  end for
end procedure

procedure BACKWARDPASS()
  for $t = T, \dots, 1$ **do**
    $(\mathcal{B}_t, v) := \text{ILMS\_UPDATE}(t)$
    $v_t^* := \max_k \{v_k\}$
    $d_t^* := \operatorname{argmax}_k \{v_k\}$ ▷ Update local discrete state
    $\overleftarrow{\mu}_t(x_{t-1}) := \int \phi_t(x_{t-1:t}, d_t{=}d_t^*)\overleftarrow{\mu}_{t+1}(x_t)\,\mathrm{d}x_t \equiv \mathbf{marg}_{x_t}\left[\phi_t^{(d_t^*)} \cdot \overleftarrow{\mu}_{t+1}\right]$
  end for
end procedure

procedure ILMS_UPDATE(t)
  for $k = 1, \dots, K$ **do**
    $\mathcal{B}^{(d_t=k)}(x_{t-1:t}) := \overrightarrow{\mu}_{t-1}(x_{t-1})\phi_t(x_{t-1:t}, d_t{=}k)\overleftarrow{\mu}_{t+1}(x_t) \equiv \overrightarrow{\mu}_{t-1} \cdot \phi_t^{(d_t=k)} \cdot \overleftarrow{\mu}_{t+1}$
    $v_k := \int \mathcal{B}^{(d_t=k)}(x_{t-1:t})\,\mathrm{d}x_{t-1:t}$
  end for
  return $(\mathcal{B}, v)$
end procedure

Figure 4.5: Iterative Local Model Selection (Simplified Case)

4.4.2 General Case

In the more general SLDS model, we must deal with dependencies between discrete variables as shown in Fig. 4.1. We can no longer pick a single value of d_t as in Eq. 4.45 without taking into account d_{t-1} since each local potential ϕ_t now contains both d_{t-1} and d_t in its domain

$$\phi_t(x_{t-1}, x_{t-1}, d_{t-1}, d_t) = \mathbf{P}(z_t \mid x_t, d_t) \cdot \mathbf{P}(x_t \mid x_{t-1}, d_t) \cdot \mathbf{P}(d_t \mid d_{t-1}) \tag{4.46}$$

The domain of the forward and backward messages is also expanded to $\overrightarrow{\mu}_t(x_t, d_t)$ and $\overleftarrow{\mu}_t(x_{t-1}, d_{t-1})$. When computing the forward message, for example, we must marginalize over x_{t-1} and d_{t-1}, but leave x_t and d_t in the domain of the message. For the backward message, we must marginalize over x_t and d_t, but leave x_{t-1} and d_{t-1} in the domain of the message.

Prior to updating each message, the general version of the algorithm uses the current best estimate of the discrete variables as a proxy for the correct marginal distribution. Because we are marginalizing over only one of the two local discrete variables, however, our best estimate must be conditional on the unspecified value of the remaining variable. This conditional estimation of the discrete variables is the same as in the first pass of the Approximate Viterbi algorithm. Given the current messages, we compute the local belief $\mathcal{B}_t(x_{t-1}, x_t, d_{t-1}, d_t)$ and construct a table of posterior likelihoods v_{ij}

$$\mathcal{B}_t^{(d_{t-1}=i, d_t=j)} = \overrightarrow{\mu}_{t-1}^{(d_{t-1}=i)} \cdot \phi_t^{(d_{t-1}=i, d_t=j)} \cdot \overleftarrow{\mu}_{t+1}^{(d_t=j)} \tag{4.47}$$

$$v_{ij} = \int \mathcal{B}^{(d_{t-1}=i, d_t=j)}(x_{t-1:t})\, \mathrm{d}x_{t-1:t} \tag{4.48}$$

To compute the forward message, we must marginalize over d_{t-1}, so we build a table of the best value of d_{t-1} for each possible value of $d_t = j$

$$d_{t-1}^*\big|_{d_t=j} = \operatorname*{argmax}_i \{v_{ij}\} \tag{4.49}$$

The forward message is computed from this table as

$$\overrightarrow{\mu}_t^{(d_t=j)} = \operatorname*{\mathbf{marg}}_{x_{t-1}} \left[\overrightarrow{\mu}_{t-1}^{\left(d_{t-1}=d_{t-1}^*\big|_{d_t=j}\right)} \cdot \phi_t^{\left(d_{t-1}=d_{t-1}^*\big|_{d_t=j}, d_t=j\right)} \right] \tag{4.50}$$

For the backward message we need to approximate marginalizing over d_t, so we compute the best value of d_t for each possible value of $d_{t-1} = i$

$$d_t^*|_{d_{t-1}=i} = \underset{j}{\text{argmax}} \left\{ v_{ij} \right\} \tag{4.51}$$

and the backward message itself as

$$\overleftarrow{\mu}_t^{(d_{t-1}=i)} = \underset{x_t}{\textbf{marg}} \left[\overrightarrow{\mu}_{t+1}^{\left(d_t = d_t^*|_{d_{t-1}=i}\right)} \cdot \phi_t^{\left(d_{t-1}=i, d_t = d_t^*|_{d_{t-1}=i}\right)} \right] \tag{4.52}$$

The resulting algorithm is the general version of the Iterative Local Model Selection shown in Fig. 4.6. The proof of convergence for this version of the algorithm is shown in Appendix A, Prop. 2, as it does not provide much additional insight.

procedure ILMS_General($\{\phi_t\}_{t=1,\dots,T}$)
  for $t = 1, \dots, T$ **do** ▷ Initialize messages
    $\overrightarrow{\mu}_t(x_t, d_t) := \mathbf{1}$
    $\overleftarrow{\mu}_t(x_{t-1}, d_{t-1}) := \mathbf{1}$
  end for
  while not converged **do**
    ForwardPass()
    BackwardPass()
  end while
  return $\left(\{\mathcal{B}_t\}, \{d_t^*\}\right)$ ▷ Return local beliefs and discrete states
end procedure

procedure ForwardPass()
  for $t = 1, \dots, T$ **do**
    $(\mathcal{B}_t, v) := \text{ILMS\_Update}(t)$
    $v_t^* = \max\{v_{ij}\}$
    $(d_t^*, d_{t-1}^*) := \text{argmax}_{(i,j)}\{v_{ij}\}$ ▷ Update local discrete state
    for $j = 1, \dots, K$ **do**
      $k := \text{argmax}_i\{v_{ij}\}$ ▷ Compute $k = d_{t-1}^*\big|_{d_t=j}$
      $\overrightarrow{\mu}_t^{(d_t=j)} := \mathbf{marg}_{x_{t-1}}\left[\overrightarrow{\mu}_{t-1}^{(d_{t-1}=k)} \cdot \phi_t^{(d_{t-1}=k, d_t=j)}\right]$
    end for
  end for
end procedure

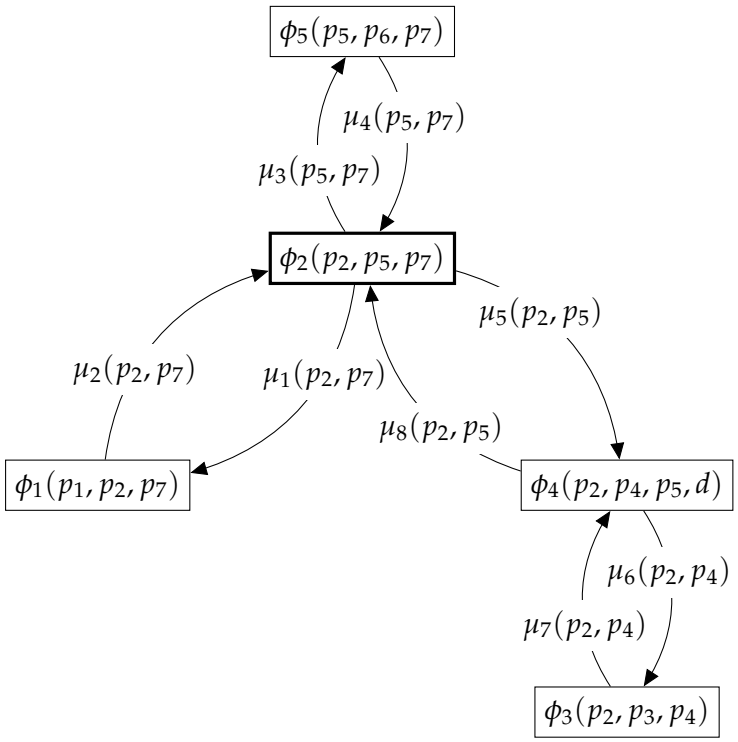
procedure BackwardPass()
  for $t = T, \dots, 1$ **do**
    $(\mathcal{B}_t, v) := \text{ILMS\_Update}(t)$
    $v_t^* = \max\{v_{ij}\}$
    $(d_t^*, d_{t-1}^*) := \text{argmax}_{(i,j)}\{v_{ij}\}$ ▷ Update local discrete state
    for $i = 1, \dots, K$ **do**
      $k := \text{argmax}_j\{v_{ij}\}$ ▷ Compute $k = d_t^*\big|_{d_{t-1}=i}$
      $\overleftarrow{\mu}_t^{(d_{t-1}=i)} := \mathbf{marg}_{x_t}\left[\phi_t^{(d_{t-1}=i, d_t=k)} \cdot \overleftarrow{\mu}_{t+1}^{(d_t=k)}\right]$
    end for
  end for
end procedure

procedure ILMS_Update(t)
  for $i = 1, \dots, K$ **do**
    for $j = 1, \dots, K$ **do**
      $\mathcal{B}^{(d_{t-1}=i, d_t=j)} := \overrightarrow{\mu}_{t-1}^{(d_{t-1}=i)} \cdot \phi_t^{(d_{t-1}=i, d_t=j)} \cdot \overleftarrow{\mu}_{t+1}^{(d_t=j)}$ ▷ Update local beliefs
      $v_{ij} := \int \mathcal{B}^{(d_{t-1}=i, d_t=j)}(x_{t-1:t})\, \mathrm{d}x_{t-1:t}$ ▷ Update scores
    end for
  end for
  return $(\mathcal{B}, v)$
end procedure

Figure 4.6: Iterative Local Model Selection (General Case)

4.4.3 Clique Tree Case

Having defined ILMS for SLDS models, we briefly discuss how the algorithm generalizes to clique trees. In order to guarantee convergence with an SLDS model, ILMS message passing must be run forwards and backwards. The requirement for a general clique tree is that messages are computed sequentially along an unbroken path through the tree. This ensures that the ILMS updates are always computed using a valid set of incoming messages. There are many possible message passing orders which satisfy this requirement, but a depth-first traversal, as shown in Fig. 4.7 satisfies the constraint. This is the ordering which will be used for the robust pose graph estimation application in Chapter 6, and a more in depth discussion is reserved until then.



(a) The modified message passing order required for ILMS. The messages, $\{\mu_i\}$, are numbered by the order in which they are computed. This order corresponds to a depth-first traversal of the tree starting from the root (bold outline). The messages form a cycle which is repeated until the algorithm converges.

1: **procedure** ComputeMessage[μ_6]
2: $\quad \mathcal{B}(p_2, p_4, p_5, d) := \phi_4(p_2, p_4, p_5, d) \cdot \mu_5(p_2, p_5) \cdot \mu_7(p_2, p_4)$;
3: $\quad d^* := \operatorname{argmax}_k \left\{ \int \mathcal{B}(p_2, p_4, p_5, d{=}k)\, \mathrm{d}p_2\, \mathrm{d}p_4\, \mathrm{d}p_5 \right\}$;
4: $\quad \mu_6(p_2, p_4) := \int \phi_4(p_2, p_4, p_5, d{=}d^*) \cdot \mu_5(p_2, p_5)\, \mathrm{d}p_5$;
5: **end procedure**

(b) The computation of μ_6 using ILMS. Marginalization over the discrete variable d is avoided using the procedure illustrated in Fig. 4.4.

Figure 4.7: ILMS message passing on the clique tree from Fig. 3.12 with a discrete variable, d, added to the domain of ϕ_4.

4.5 Evaluation

We evaluate two versions of the ILMS algorithm alongside the deterministic methods described in Section 4.3:

- `ILMS_ML`: Iterative Local Model Selection using Eq. 4.44
- `ILMS_MM`: Iterative Local Model Selection using Eq. 4.45
- `AV`: Approximate Viterbi[99]
- `GPB2`: Generalized Pseudo-Bayesian Order 2 Filter [97]
- `GPB2S`: Generalized Pseudo-Bayesian Order 2 Smoother [97]
- `VB`: Variational Bayes [103]
- `EP`: Expectation Propagation [109]

All algorithms were implemented in MATLAB. We include both GPB2 and GPB2S because the smoothing version of the algorithm fails when the assumption underlying its approximation is violated. In this case the filtering version performs better and is used as a proxy.

The different methods are evaluated on a set of three synthetic datasets with increasing difficulty: `OUT`, `MNV1`, and `MNV2`. The `OUT` dataset simulates a robust estimation problem with an observation model allowing for outliers. The other two datasets simulate a maneuvering target with various dynamics. All three datasets are based on the following SLDS model

$$d_t \in \{1, \ldots, K\}, x_t \in \mathbb{R}^N, z_t \in \mathbb{R} \tag{4.53}$$

$$d_1 \sim \mathcal{D}(p_0) \tag{4.54}$$

$$d_t \,|\, d_{t-1}{=}i \sim \mathcal{D}(\tau_i) \qquad t > 1 \tag{4.55}$$

$$x_1 \sim \mathcal{N}(\mathbf{0}, \Sigma_0) \tag{4.56}$$

$$x_t \,|\, x_{t-1}, d_t{=}j \sim \mathcal{N}\left(A^{(j)} x_{t-1} + b^{(j)}, \Sigma^{(j)}_{\text{mot}}\right) \qquad t > 1 \tag{4.57}$$

$$z_t \,|\, x_t, d_t{=}j \sim \mathcal{N}\left(G x_t, \Sigma^{(j)}_{\text{obs}}\right) \tag{4.58}$$

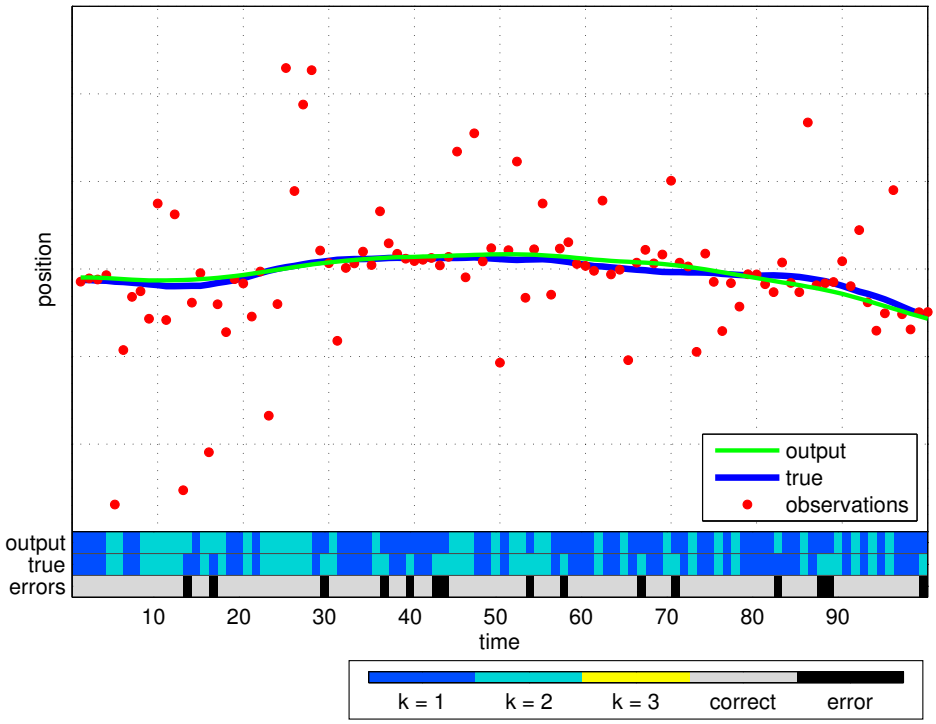


Figure 4.8: A sample from the `OUT` experiment with output from the proposed `ILMS_MM` algorithm.

The number of switching states in the system is controlled by the integer K, and the size of the continuous state space by N. A listing of the parameter values used for the evaluations is shown in Tab. A.1. The notation $d \sim \mathcal{D}(p)$ describes a discrete distribution with $\mathbf{P}(d{=}i) = p_i$.

In the first experiment, `OUT`, the observation model consists of a mixture distribution between low covariance observations and high covariance outliers. This simulates the problem of automatically detecting and discarding outlier measurements using two switching modes ($K = 2$). If $d_t = 1$, a low amount of sensor noise is assumed ($\Sigma_{\text{obs}}^{(1)}$ small). When $d_t = 2$, the sensor noise is high to simulate an outlier measurement very loosely coupled with the true behavior of the system ($\Sigma_{\text{obs}}^{(2)}$ large). In this experiment, the motion model does not depend on the switching variable. Since the outliers are uncorrelated in time, d_t is independent of d_{t-1} and so $\tau_1 = \tau_2 = p_0$. Fig. 4.8 shows a sample from the `OUT` experiment.

`MNV1` models a maneuvering target with three different behavior regimes. In this case the dynamics of the system also change based on the discrete state. This experiment simulates a target which moves in roughly straight lines, but occasionally makes large changes in direction and occasionally slows down. Three possible values of the discrete state d_t control this behavior

- **Standard ($d_t = 1$):** The motion model is constant velocity with a small amount of noise in the velocity term, resulting in motion characterized by gradual changes in direction.
- **Maneuvering ($d_t = 2$):** A large amount of noise in the velocity term allows quick changes of direction. More noise is also added to the observation model to simulate a regime where the sensor is less accurate.
- **Braking ($d_t = 3$):** In the final mode, the target is braking/stopped. The motion model halves the velocity and adds almost no noise to the velocity term in between time steps. The observation noise is identical to the $d_t = 1$ case.

Fig. 4.9 shows a sample from the `MNV1` experiment.

`MNV2` models a maneuvering target loosely based on a bouncing ball under the influence of various forces. The ball falls under constant acceleration, but occasionally is lifted upward by an external force. In addition, a third mode allows the ball to 'bounce' by randomly switching the sign of its velocity. This experiments uses a constant acceleration motion model and so has a three dimensional state space including position, velocity, and acceleration.

- **Falling ($d_t = 1$):** A constant negative acceleration is applied to the velocity.
- **Lifting ($d_t = 2$):** A constant positive velocity is applied to the position.
- **Bouncing ($d_t = 3$):** An instantaneous change of direction flips the sign of the velocity term; this model cannot be active for two consecutive times.

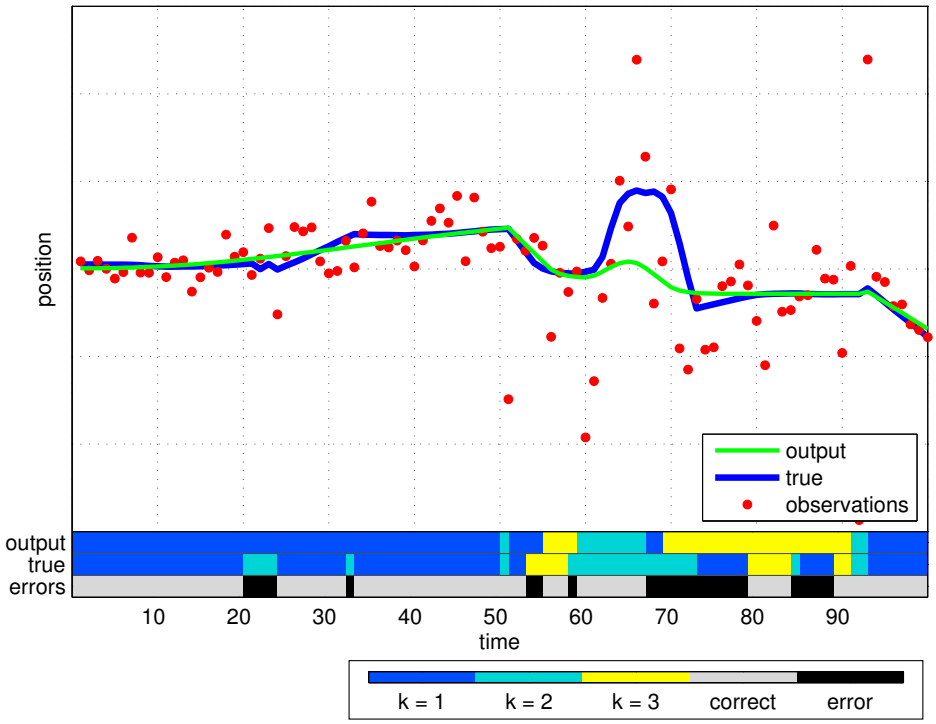


Figure 4.9: A sample from the `MNV1` experiment with output from the proposed `ILMS_MM` algorithm.

Fig. 4.10 shows a sample from the `MNV2` experiment.

For each experiment, 100 random samples were drawn from the model in order to evaluate the algorithms. Root mean square error of the estimated position was used to measure estimation accuracy in terms of the continuous variables. Estimation accuracy for the discrete variables was measured via the discrete error rate, which is defined as the fraction of incorrectly estimated discrete states. We note that the continuous error rate is more significant because the discrete states are not always uniquely identifiable. The results in terms of both discrete and continuous errors are shown Fig. 4.11. The corresponding execution time statistics are shown in Tab. 4.1. Side-by-side comparisons of the outputs of the evaluated algorithms are shown in Appendix A as Fig. A.1, Fig. A.2, and Fig. A.3. In addition, we ran another set of experiments from the same models with the standard deviation of the observation noise scaled by $\kappa \in \{0.05, 0.2, 1.0, 5, 20\}$. An additional 50 samples were generated for each value of κ and plots of the median error as a function of observation noise scale was generated. These plots are also shown in Appendix A as

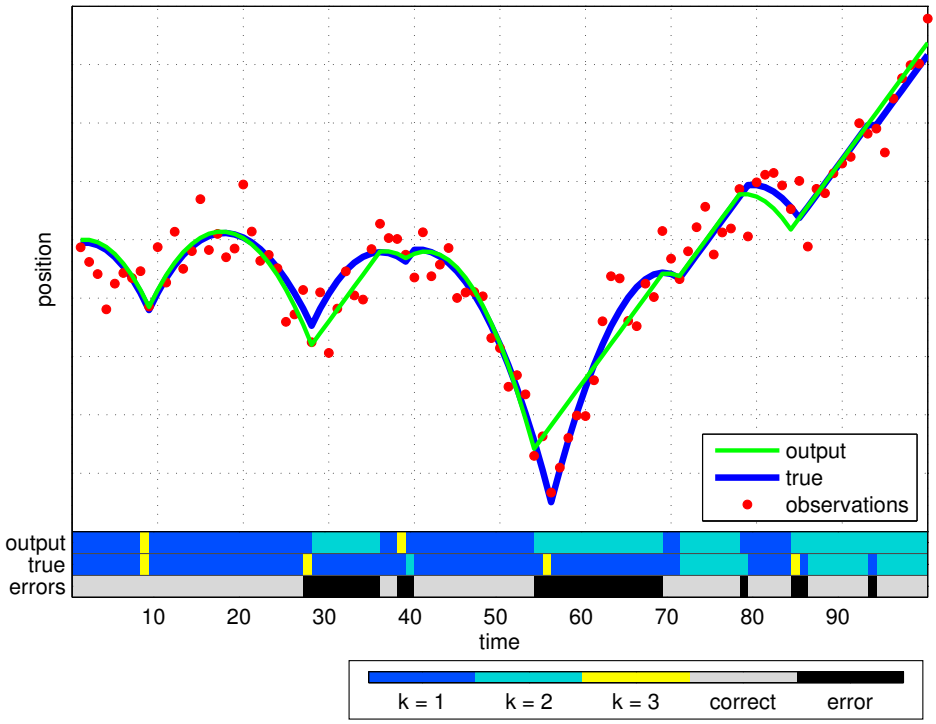


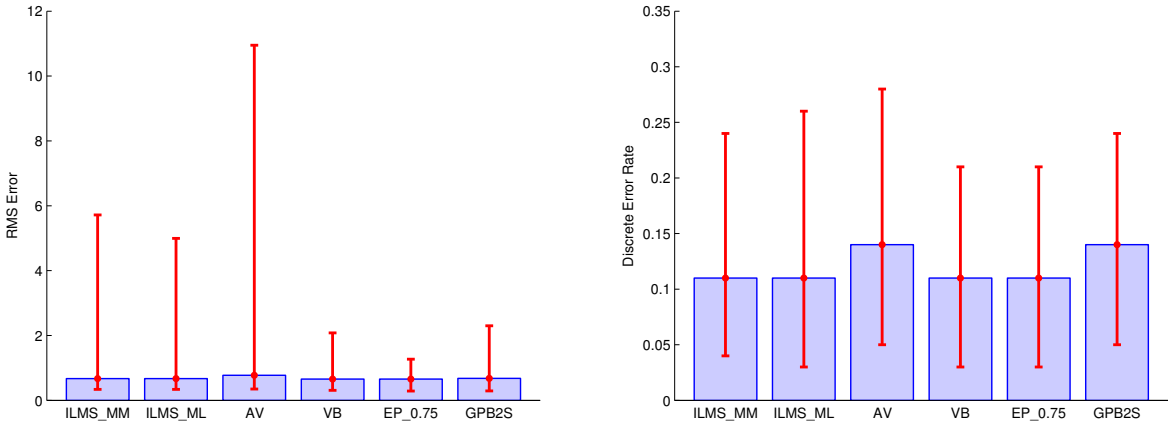
Figure 4.10: A sample from the MNV2 experiment with output from the proposed ILMS_MM algorithm.

Fig. A.4.

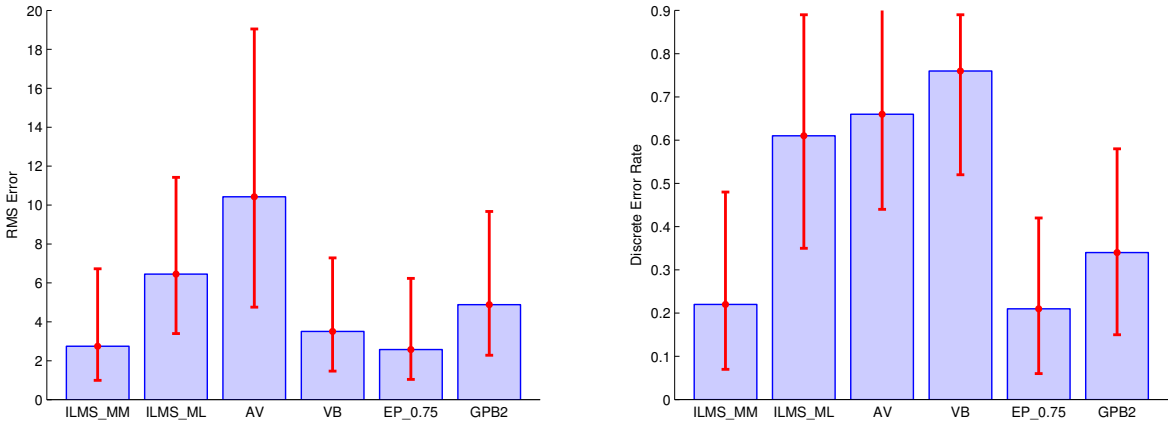
Algorithm	min (s)	median (s)	mean (s)	max (s)
ILMS_MM	0.96	1.97	2.11	7.00
ILMS_ML	0.87	1.75	2.00	6.22
AV	0.32	0.54	0.45	0.58
VB	6.51	9.85	10.04	28.52
EP	7.27	9.77	14.55	33.85
GPB2S	0.16	0.28	0.23	0.36

Table 4.1: The mean, median, min, and max running times over all experiments for the algorithms evaluated.

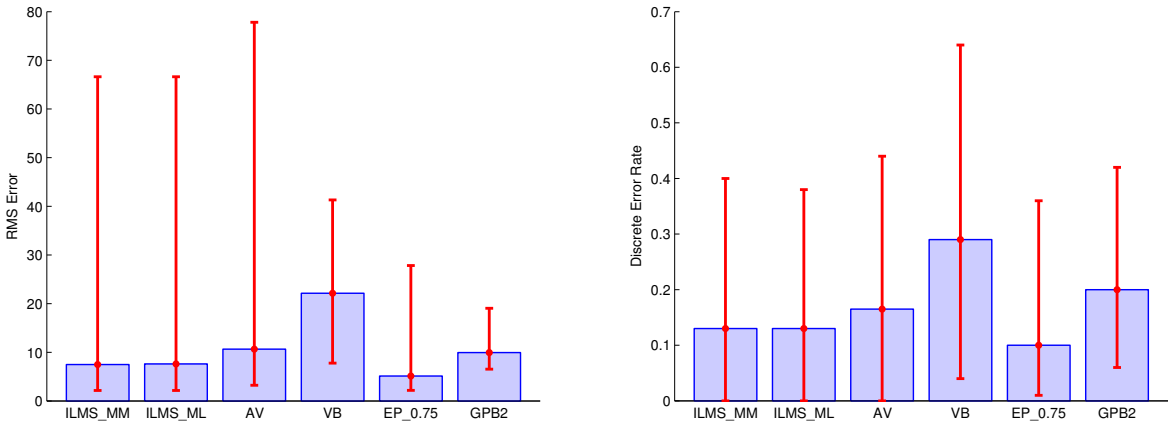
In the OUT experiment, we can see that all of the algorithms are roughly equivalent, with AV having notably bad worst-case RMSE. The identification of uncorrelated outliers appears to be a relatively easy task for all of the algorithms, but does serve a role as an initial validation of ILMS. We note that in this robust estimation role, the ILMS_ML version of our algorithm does not perform any worse than ILMS_MM. This will come in handy for Robust Pose Graph Estimation in Chapter 6, where it is more natural to use a MAP procedure.



(a) `OUT`

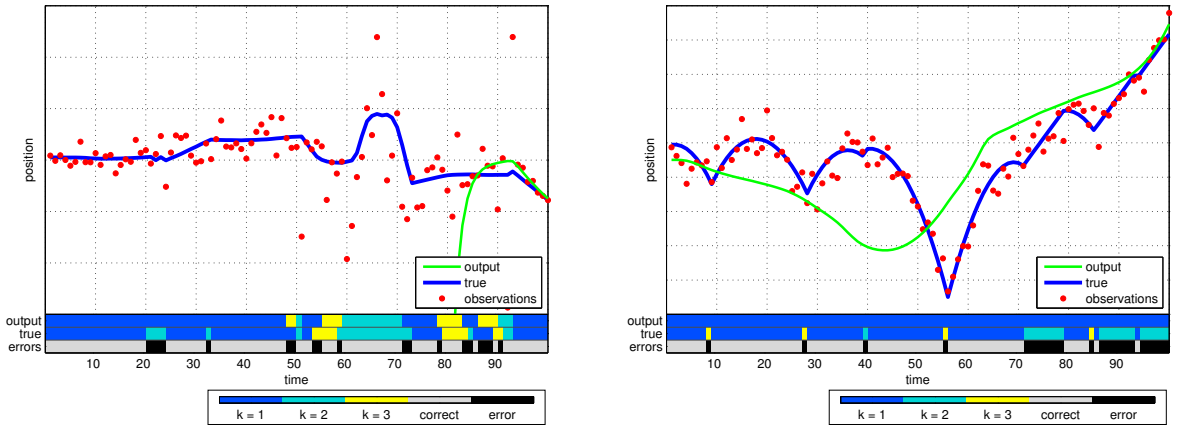


(b) `MNV1`



(c) `MNV2`

Figure 4.11: Results for the three experiments showing median RMS error in the estimated position (left) and the discrete error rate (right). The error bars indicate the maximal and minimal values.

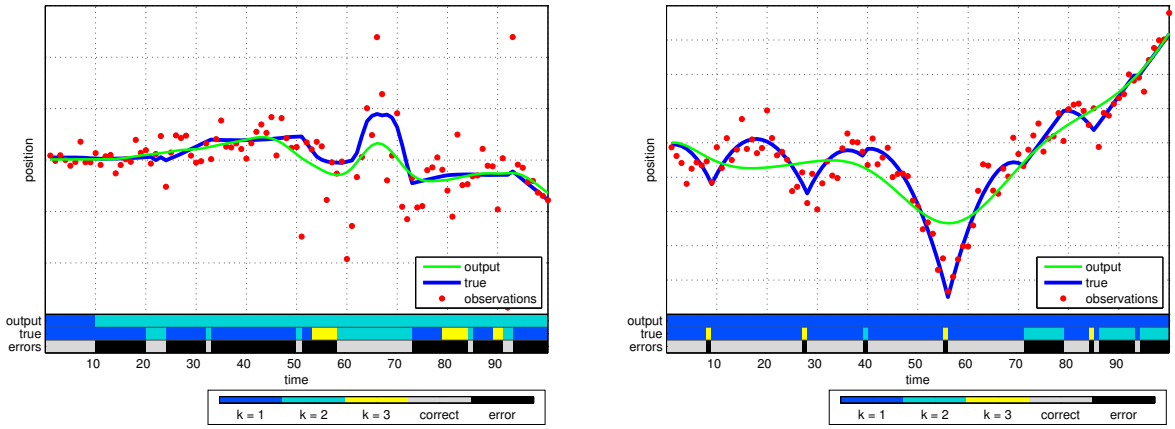


(a) MNV1 **(b)** MNV2

Figure 4.12: Characteristic failures of the GPB2S algorithm observed in experiments MNV1 and MNV2.

The MNV1 experiment is more difficult, and in this case we can see that the ILMS_MM algorithm performs better than ILMS_ML and all other algorithms except EP, which performed extremely well. In this experiment we observed a general failure of the GPB2S algorithm, which performed even worse than its filtering counterpart. This appears to be a failure in the basic approximating assumption of the algorithm which results in 'incorrect' models being averaged into the estimate during the backward pass. Fig. 4.12 shows the results of running GPB2S on the sequence shown in Fig. 4.9. It is also noteworthy that while VB performed well in terms of RMSE, it had the worst performance in terms of discrete error rate. This appears to be caused by the averaging of different models. Despite the wrong model having the most weight, it is still possible to get a reasonable estimate of the continuous variables as shown in Fig. 4.13.

MNV3 is the hardest of the three experiments. It contains more complex target dynamics, rapid changes of direction, and strong biases in the motion models combined with an all around high level of observation noise. This creates a situation where using the correct motion model results in drastically better state estimates. This intuition is confirmed by Fig. 4.11c where we can see a strong correspondence between discrete and continuous error rates. The performance of the algorithms follows the same general trend observed before, with EP the most accurate and ILMS_MM coming in second best.



(a) MNV1 **(b)** MNV2

Figure 4.13: Failures of the VB algorithm on MNV1 and MNV2. Despite estimating the wrong discrete model, Variational Bayes can average over all the models to obtain an 'OK' continuous estimate.

In general, we note that that performance of the EP algorithm was highly dependent on the step size schedule with larger step sizes corresponding to both better performance and a higher chance of failure. The results shown came at the expense of a failure rates of 2%, 3%, and 38% on experiments OUT, MNV1, and MNV2 respectively.

4.6 Conclusions

In this chapter we have described the Iterative Local Model Selection Algorithm for approximate inference in SLDS models. Two variants of the algorithm (ILMS_ML and ILMS_MM) were proposed based on different model selection criteria. These new algorithms are compared to existing deterministic techniques using three synthetic tracking experiments. We found that in general the ILMS_MM variant of the proposed approach is more accurate than ILMS_ML.

Overall, we found Expectation Propagation to be the most accurate when it works. Unfortunately the EP algorithm does not always converge and is the slowest method evaluated. The ILMS_MM variant of our algorithm is almost as accurate as EP, but comes with a convergence guarantee. In terms of speed, it is approximately five times faster than EP and Variational Bayes.

This combination of performance characteristics makes ILMS an interesting choice for large scale applications – it is fast enough to be applicable, and accurate enough to be useful. As will be shown in Chapter 6, ILMS can also be generalized to work on Conditional Linear Gaussian Networks, widening its applicability to the large scale estimation problems often seen in the SLAM community. The simplicity of the local maximization procedure will also be shown to be an advantage as it makes it possible to substitute more interesting discrete optimization procedures as part of the local model selection step. In Chapter 5 we will make use of this by computing data associations as the model selection procedure within ILMS.

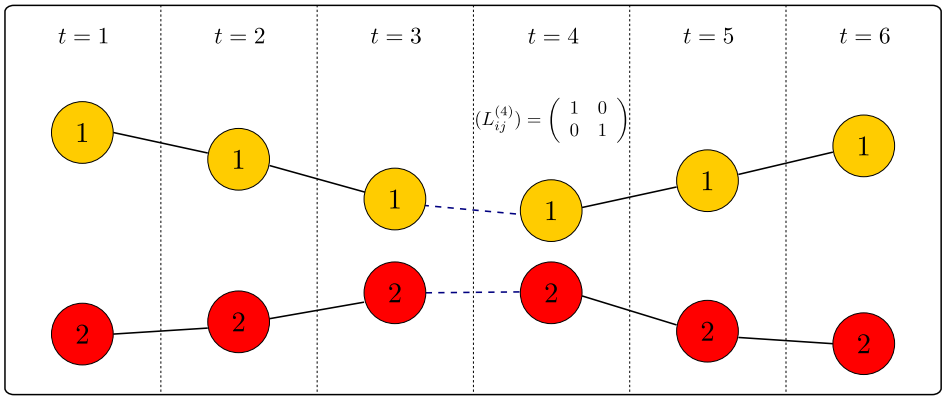
5

Multi-Target Tracking

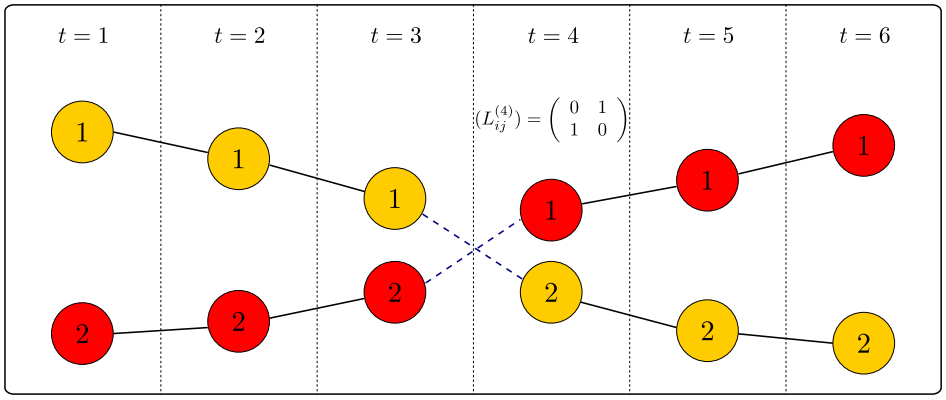
5.1 Introduction

The previous chapter introduced the Iterative Local Model Selection inference strategy and validated the basic approach. In this chapter we look at how this strategy can be applied to the problem of data association and outlier rejection in multi-target tracking. We introduce the *Latent Data Association* algorithm, which is the combination of the ILMS inference strategy with a specific parameterization of the data association problem. The data association variables are treated as the discrete variables in a *Switching Linear Dynamical System* (SLDS) [108]. The name of the algorithm is due to the fact that data associations are performed in the latent variable layer of the SLDS.

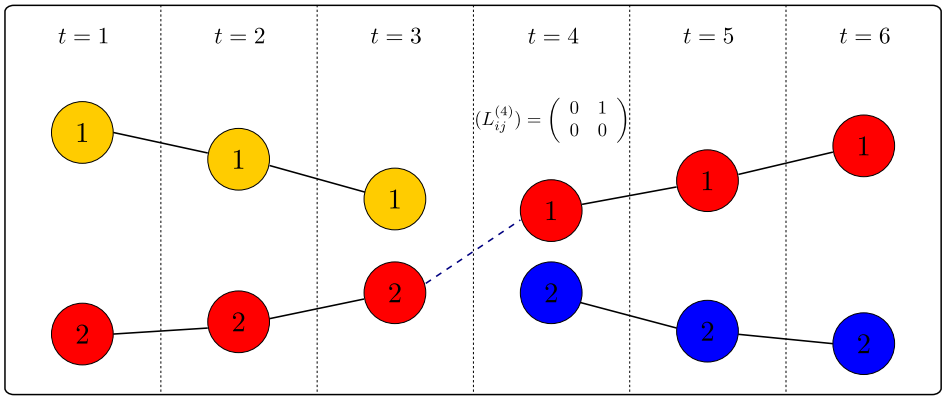
By looking at multi-target tracking as a discrete/continuous inference problem, more complex reasoning about object classification can be applied. In this spirit, we take advantage of advances in object detection and classification [118, 119, 120, 121] by incorporating object/target classification directly into our system. This is accomplished by adding discrete object category variables into the tracking model. The outputs of a standard object detector can then be used as observations of the target's category. Using this model allows the classification and tracking problem to be naturally combined into a single system.



(a) Assignment #1



(b) Assignment #2



(c) Assignment #3

Figure 5.1: Illustration of three possible Latent Data Association assignments at $t = 4$. The binary indicator matrix $(L_{ij}^{(4)})$ controls the matching of nodes between $t = 4$ and $t = 3$. Nodes are numbered within each time slice and colored based on their global track membership. Each node represents a single latent track state together with any observations (if they exist).

5.2 Related Work

Multi-target tracking is an important, but stubborn problem in Computer Vision as well as many related fields (notably robotics). The applications range from surveillance, through autonomous navigation, to active scene modeling and understanding. Despite the numerous motivations for solving this problem, it has remained a challenging topic after decades of active research. Historically, it has been difficult for two reasons. The first is the combinatorial space of possible associations between the observations and objects being tracked, and the second is model selection over the number of existing tracks. The parameterization used in this chapter combines both of these model selection problems into an SLDS model which can be solved with ILMS.

The traditional techniques for multi-target tracking and data association (i.e. JPDAF, MHT, and JCBB) have already been reviewed in Chapter 2. More recently, an approach known as *Tracking-by-Detection* [122, 123] has become popular. Tracking by Detection reframes multi-target tracking as the fusion of an object detector[118, 120, 121] with data association. In contrast to traditional methods focusing on radar data with point measurements, the *Tracking-by-Detection* literature has focused on tracking objects in video sequences. Out of the recent work, several broad strategies can be identified.

5.2.1 Probabilistic Occupancy Maps

In Probabilistic Occupancy Maps (POMs), detection probabilities are accumulated into a discretized grid over the workspace. The tracking question is formulated as linking compatible detections on the grid into consistent trajectories for each object.

Berclaz et al. [124], Fleuret et al. [125] use Dynamic Programming to find consistent paths through the POM accumulated from detections in a multi-camera setup. Each track is processed individually with heuristics used to resolve conflicting data associations. Appearance information is fused into the algorithm to enable robust data association over long

sequences. As a downside, heuristics must still be used in order to resolve data association conflicts between the tracks since the Viterbi algorithm is applied independently for each object, as well as to estimate the total number of targets to be tracked.

Andriyenko and Schindler [126] proposes a global optimization approach which considers all tracks together and avoids the heuristics of [124, 125]. This is done by formulating the problem as the relaxation of an Integer Linear Program (ILP) for approximate global optimization. The authors also show how appearance information as well as target dynamics can be included into the POM formulation of tracking.

Berclaz et al. [127] builds a graph over every possible discrete location at every possible time. An LP is used to formulate a max-flow problem where each vertex contains a maximum of one object, and flow between adjacent nodes at time t and $t-1$ model targets moving between these nodes. Virtual nodes are used to allow objects to enter and leave the scene. For efficiency, a sparse version of this graph is constructed – unreachable vertices are pruned from the graph.

Berclaz et al. [128] also constructs a graph over the set of discrete object positions on a grid, but formulates the tracking problem as a K-shortest paths problem in order to obtain a fast global solution. Although they do not include an appearance model, a later extension[129] shows how an appearance model based on per-part color histograms can be incorporated into the same framework.

Discretization of the tracking space limits applications of POM based trackers because the technique cannot easily be applied to a moving sensor platform. The discretization also forces a compromise between accuracy and the size of the tracking area. Unlike these approaches, we do not make any discretization; continuous variables are treated as such and smoothing of the output trajectories is performed via the motion model without any post-processing.

5.2.2 Detection Partitioning

The second direction, which we refer to as *Detection Partitioning*, avoids discretization by partitioning the set of detections into distinct groups corresponding to target tracks. Once detections are separated into tracks, the trajectory of each target cat be recovered by smoothing and interpolating across any missing observations.

Wolf et al. [130] is an early work using dynamic programming to find the most likely linking of detections into tracks. The jointly optimal solution for all tracks is found simultaneously, unlike other approaches (e.g. [124, 125]). Unfortunately, an unrealistically low missing detection rate is required.

Jiang et al. [131] form a graph of object detections, similar to the dynamic programming approach in [130]. Each object detection is a node in the graph, with the goal being to link the nodes to form independent target tracks. Unlike other DP-based methods, however, interactions between different targets are modeled as mutual exclusion constraints over the nodes. A relaxed LP formulation is presented whose solution gives the optimal target tracks. The algorithm assumes a fixed number of tracks, and so cannot be used to estimate this number directly.

Leal-Taixe et al. [132] form a graph out of the detections with edge costs representing the cost of data associations. The minimum cost flow through the resulting graph (from a virtual start node to a virtual end node) determines tracks. The number of targets is automatically inferred based on the total amount of flow. Notably, a social force model is also included.

Brendel et al. [133] builds a graph where each node corresponds to a pair of detections across adjacent frames. The nodes are weighted with a compatibility score based on the similarity of the two underlying detections. Edges connect tracklets which violate mutual exclusion constraints, allowing data association to be phrased as a maximum-weight independent set problem. This problem is re-solved iteratively, interleaved with an online

learning procedures which improves the compatibility scoring.

Detection Partitioning approaches generally assume fully observable target position and manually smooth the trajectories as an afterthought. Our approach falls within this framework, but keeps the hidden state space model, allowing for smoothing as a natural consequence of the observation model.

5.2.3 Discrete/Continuous Optimization

Leibe *et al.* [134, 90, 135, 136] propose a batch method which combines detections from the Implicit Shape Model [137] with a tracking framework. Independent trajectory hypotheses are formed by starting with each detection and searching for the most likely completions both forward and backward in time. This over-complete set of (possibly contradictory) tracking solutions is then pruned down to the most likely non-contradictory set using a Quadratic Boolean Program.

While a number of the Tracking-by-Detection approaches rely on exact minimization of an approximate cost function, others have focused on local optimization of a more accurate energy function [138, 139]. In Andriyenko and Schindler [138], a continuous, non-convex objective function is developed which includes soft mutual exclusion constraints. Andriyenko et al. [139] extends the former with a combined discrete data association phase formulated as a label-assignment problem over the detections. The authors show that decreases in the objective energy correspond well with increases in performance as measured by various metrics.

The discrete-continuous optimization formulation of tracking is close to our method in spirit. In our case, however, tracking is formulated as an inference rather than optimization problem. As a general contrast to the optimization viewpoint, our method is probabilistically motivated as an approximation to the Bayesian posterior. This has the advantage of meaningful parameters and confidence scores on the output.

5.2.4 Monte Carlo

Monte Carlo based approaches are both principled and simple to implement even for complicated non-linear models. They rely on representing a distribution over the state space as a collection of discrete samples. In the case of Particle Filters (PF), these samples are manipulated so that their distribution tracks the posterior of the filter as a function of time. Markov Chain Monte Carlo (MCMC) can be used as part of a particle filter to generate samples from an otherwise intractable distribution, or as an independent tracking algorithm by sampling over the joint posterior of the whole problem. In practice, naive Monte Carlo implementations can run into problems when the state space increases beyond two or three dimensions as the number of samples required tends to grow exponentially with state space dimension. In the case of MCMC, convergence is also notoriously difficult to diagnose if accurate samples from the posterior are required.

There are several sampling strategies and techniques which have been developed to adapt particle filters to the problem of multi-target tracking. One of the first practical applications, Schulz et al. [84], combined a particle filter with the Joint Probabilistic Data Association Filter (JPDAF) to track people from a mobile platform using 2D laser data. Using a particle filter allowed the data association weights of the JPDAF to be computed efficiently by summing over particle weights. Since the JPDAF requires manual initialization of tracks, a separate filter was used to track the number of targets in the scene and manually add or remove targets. Vermaak et al. [140] present a generalization of this idea to include multiple observers and arbitrary proposal distributions.

Doucet et al. [141] introduced an alternative filter which attempts to sample from the full Bayesian posterior over the number of targets, the data associations, and the individual target states. Sampling from the resulting Switching Markov Model requires an Auxiliary Particle Filter[142] combined with an Unscented Transform[20] approximation. In general, particle filters tend to poorly represent the multi-modal posterior distributions which sometimes arise in multi-target tracking. To address this limitation, Vermaak et al. [143]

propose a mixture of particle filters where each component can track a separate mode of the posterior.

Because sampling techniques allow both non-linear and hybrid discrete-continuous likelihood models, they open the door to incorporating additional information and more complicated interaction models. For example, Giebel et al. [144] use particle filters to track deformable objects in 3D by integrating information from multiple data sources including appearance models, object detections, and stereo data. Khan et al. [145, 146] propose a tracker which automatically manages the number of targets and allows interactions between individual targets. In this case, Reversible Jump Markov Chain Monte Carlo[147] was used as part of the particle filter in order to sample both the number of targets and the data associations.

Cai et al. [148] combines Tracking-by-Detection with a mixture of particle filters. The output of the detector is incorporated into the proposal distribution in order to improve the efficiency of the sampler. Data association is computed by solving a linear assignment problem between currently existing targets and observations. Unlike the data association parameterization proposed in this chapter, however, their algorithm requires manual addition and removal of targets based on heuristics. Breitenstein et al. [149, 150] propose an alternative Tracking-by-Detection particle filter which also incorporates detector scores as a measure of confidence.

Random Finite Sets [151, 152] are a proposed alternative probabilistic calculus designed specifically for dealing with finite sets of targets. Here, a specialized theory is developed for treating a dynamically sized set of target states as a single random variable to be tracked. This is perhaps the most principled approach to multi-target tracking, but requires a specialized set of mathematical tools. Our method offers some of the same advantages, but stays within the 'standard' probabilistic framework.

Aside from filtering, Monte Carlo can also be used as a smoother. Oh et al. [32] propose an MCMC system which samples over partitions of the detections. Each sampling step in

their algorithm involves running a Kalman Smoother to solve for the posterior distribution of the continuous state space conditioned on the proposed detection partitions.

More recently, Benfold and Reid [153] also used MCMC to partition detections, but removes the hidden continuous state space entirely. Instead, they used a simplified tracking model where the target positions are fully observed at each detection (as is common in many of the previously mentioned Tracking-by-Detection approaches). In addition, KLT tracklets[154] are incorporated to help improve the accuracy of data association and generate stable bounding box tracks. Combined with a sliding window filter, this strategy allows a real-time system capable of running on high definition video.

5.3 Traditional Data Association

We review the traditional formulation as motivation for the *Latent Data Association* algorithm introduced in the following section. For the sake of simplicity, we assume a fixed number of tracks. Consider a set of observations $Z = \{Z^{(1)}, \ldots, Z^{(T)}\}$ with $Z^{(t)} = \{z_1^{(t)}, \ldots, z_{N_t}^{(t)}\}$ and t denoting time. Depending on the problem, each observation z_{ti} could include 2D/3D target locations as well as target size and other properties. These observations are assumed to be generated by M distinct targets. Each target, $m \in \{1, \ldots, M\}$ follows the trajectory $X_m = (x_m^{(1)}, \ldots, x_m^{(T)})$. The data association problem is traditionally formulated as finding a correspondence between the targets and observations at each point in time. This is done by introducing a set of discrete decision variables, $D = \{D^{(1)}, \ldots, D^{(T)}\}$, with $D^{(t)} = \{d_i^{(t)}\}$, which control the associations. In this notation, $d_i^{(t)} = j \in \{1, \cdots, M\}$ indicates that the observation $z_i^{(t)}$ is associated with the j^{th} target, with the constraint that no two observations can be assigned to the same target. The value $d_i^{(t)} = 0$ indicates an outlier observation not associated with any particular target. The graphical model for this problem is shown in Fig. 5.2a for reference.

If D is known, it is possible to infer the posterior trajectories, $\mathbf{P}\left(x_m^{(1:T)} \,\middle|\, D, Z\right)$, using M independent Kalman Smoothers. With D unknown, however, we are forced to consider all

possible data associations. This can be formulated as a posterior

$$\mathbf{P}(X_m \mid Z) = \sum_D \mathbf{P}(X_m \mid D, Z)\, \mathbf{P}(D) \tag{5.1}$$

or as a MAP problem

$$X_m^* = \underset{X_m, D}{\operatorname{argmax}}\, \mathbf{P}(X_m \mid D, Z)\, \mathbf{P}(D) \tag{5.2}$$

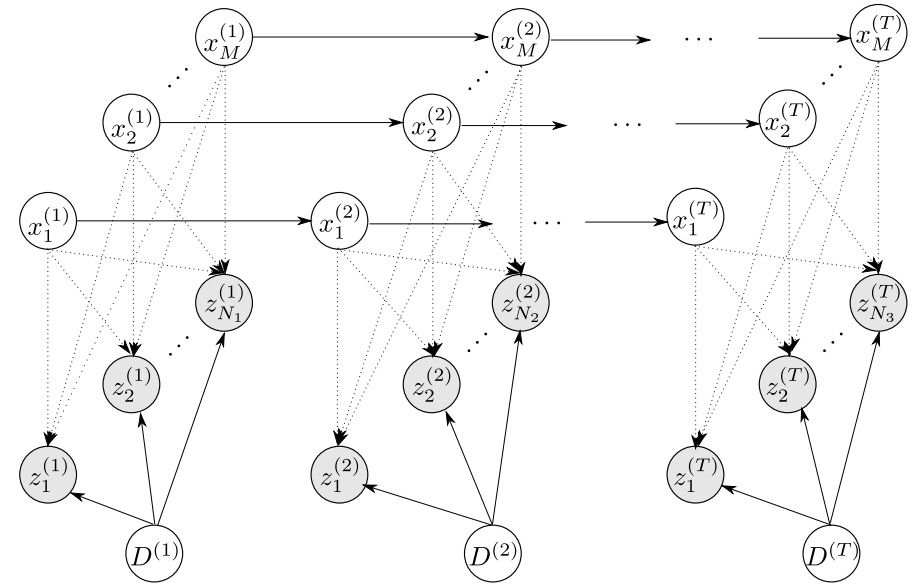
In either case, an approximation must be made to deal with the combinatorial number of possible values for D. Various search strategies exist for finding a 'good' D, but these are often prone to local minima.

Even if we were to avoid enumerating all values of D in the above, 'proper' Bayesian model selection over the number of tracks, M, still requires this enumeration because the posterior likelihood is given by

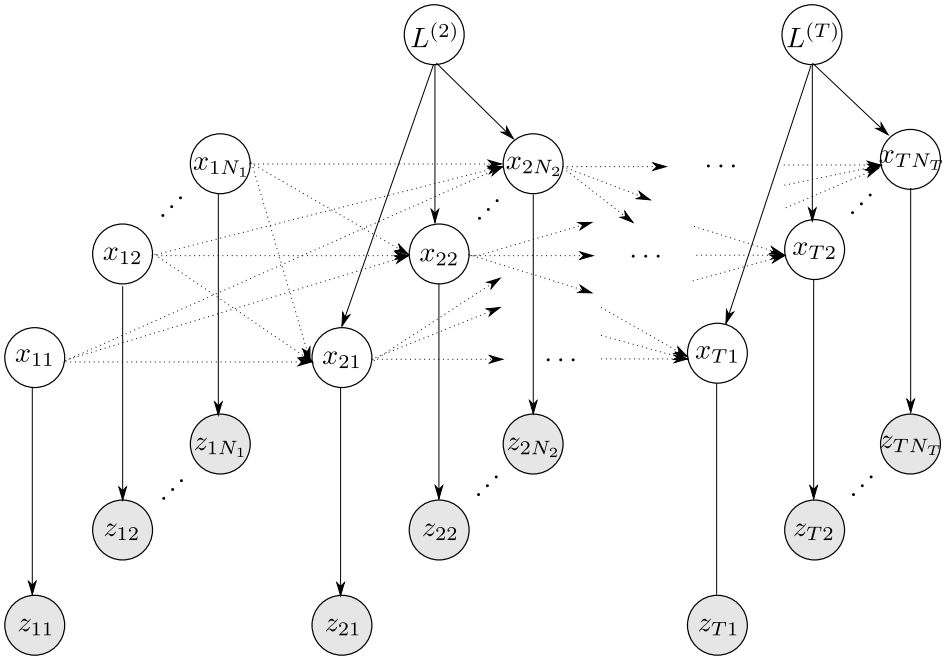
$$\mathbf{P}(M \mid Z) = \frac{\mathbf{P}(Z \mid M)\, \mathbf{P}(M)}{\mathbf{P}(Z)} \propto \tag{5.3}$$

$$\propto \mathbf{P}(M) \sum_D \int \mathbf{P}(Z \mid X, D)\, \mathbf{P}(X \mid M)\, \mathrm{d}X \tag{5.4}$$

Whereas for a fixed M we can avoid the enumeration by restricting ourselves to a MAP estimate and local optimization, the same approach cannot be used for model selection. To calculate the probability of a given value of M, we must consider the likelihood of all possible data associations conditioned on the existence of exactly M targets.



(a) Traditional data association



(b) Latent Data Association

Figure 5.2: Graphical models contrasting Latent Data Association with the traditional approach. Dashed lines represent dependencies controlled by the data association variables $D^{(t)}$ or latent data association variables $L^{(t)}$ respectively.

5.4 Latent Data Association

The *Latent Data Association* parameterization avoids the difficulties of the previous section. While the traditional approach attempts to assign observations to previously existing tracks, *Latent Data Association* starts by assuming that each detection is its own track of length one. The problem of tracking then becomes a question of linking these singleton tracks into longer trajectories. We do this by assigning each track at time t as the continuation of some track at $t-1$. This amounts to a set of discrete variables controlling how to join the tracks after time t with those existing up to time $t-1$. We refer to this form of data association as *latent* because the discrete variables now control associations between adjacent latent state variables. Fig. 5.1 illustrates this parameterization with the tracks being spliced between $t=3$ and $t=4$.

To define this model formally, we define a *node* as the set of state variables at a specific time instance as well as any observations of this state. Each node is denoted by the pair $n=(t,i)$, where t is the time index and i an index within that time slice (illustrated in Fig. 5.1). For the node $n=(t,i)$, we define x_{ti} as the unobserved state variables of the node and z_{ti} as the observations (if present). A node without any associated observations is referred to as 'virtual'.

The binary indicator matrix $L^{(t)}$ is used to control the latent data associations at time t; setting $L_{ij}^{(t)}=1$ corresponds to linking node (t,i) with node $(t-1,j)$. If $\forall j,\ L_{ij}^{t}=0$, we know that node (t,i) is not linked with anything in the past and hence represents the start of a new track. In order to ensure track continuations are always one-to-one, we must enforce the mutual exclusion constraints $\sum_i L_{ij}^{(t)} \leq 1$ and $\sum_j L_{ij}^{(t)} \leq 1$.

Given these definitions, the set of nodes combined with a value for each $L^{(t)}$ matrix forms a graph structure, seen in Fig. 5.1 , where each connected component represents an independent track. This parameterization of the problem subsumes standard data association as well as model selection over the number of tracks; any number of tracks and any

data association can be represented with a suitable value for $L = \{L^{(t)}\}$.

By fixing the set of latent data association indicators, we partition the nodes into independent tracks. Within each such track, we have the standard motion and observation models. Each observation z_{ti} is generated from the associated target state x_{ti} according to an observation model, $\mathbf{P}(z_{ti} \mid x_{ti})$. The motion model between any two nodes is specified conditional on these nodes being connected:

$$\mathbf{P}\left(x_{ti} \,\middle|\, x_{t-1,j}, L_{ij}^{(t)}=1\right) \tag{5.5}$$

The associated graphical model is shown in Fig. 5.2b.

If we assume linear motion and observation models, the model forms an SLDS[108] where the discrete $L^{(t)}$ variables control the relationships between continuous variables in the Markov Chain. This SLDS can be used to implicitly solve the data association problem together with model selection over the number of targets.

On their own, neither the use of the linear assignment problem for data association, nor the concept of associating detections in time are novel. Among others, Cai et al. [148] explicitly made use of the Linear Assignment Problem for track-to-observation data associations. Similarly, many of the Detection Partitioning algorithms discussed in Section 5.2.2 operate by grouping detection using linear programming and hence can automatically infer the number of targets[130, 131, 132, 133]. In order to do so, however, this latter set of algorithms is forced to forego the 'hidden' layer in the traditional tracking model. The novelty of the Latent Data Association approach lies in combining these ideas with the ILMS algorithm, which allows us to reason about the entire space of hidden continuous variables for each track while computing the data associations. Computing the associations horizontally in time, rather than vertically to the detections, allows the number of targets to be inferred as part of the same procedure.

5.5 Approximate Inference with ILMS

We use the Iterative Local Model Selection algorithm described in Chapter 4 to solve the SLDS introduced in the previous section. The goal is to pick $L^* = \text{argmax}_L \, \mathbf{P}(Z \mid L)$ and compute the smooth trajectories $X^* = \text{argmax}_X \, \mathbf{P}(X \mid Z, L^*)$. This corresponds to the ILMS_MM algorithm from Chapter 4.

If the value of L were known, the problem would be reduced to smoothing trajectories based on the partitioned observations. We use the message passing notation to describe the smoothing process with L fixed. For a node (t, i), we define $\texttt{pr}_{ti}$ as the index of the previous node (at $t-1$) in the same track and $\texttt{nx}_{ti}$ as the index of the next node (at $t+1$). As a shorthand, we also define $x_{ti}^{\texttt{pr}} \equiv x_{t-1,\texttt{pr}_{ti}}$. The messages can then be defined recursively as

$$\phi_{ti}(x_{ti}^{\texttt{pr}}, x_{ti}) \equiv \mathbf{P}(z_{ti} \mid x_{ti}) \cdot \mathbf{P}\big(x_{ti} \mid x_{ti}^{\texttt{pr}}\big) \tag{5.6}$$

$$\overrightarrow{\mu}_{ti}(x_{ti}) := \underset{x_{ti}^{\texttt{pr}}}{\textbf{marg}}\big[\overrightarrow{\mu}_{t-1,\texttt{pr}_{ti}} \cdot \phi_{ti}\big] \tag{5.7}$$

$$\overleftarrow{\mu}_{ti}(x_{ti}^{\texttt{pr}}) := \underset{x_{ti}}{\textbf{marg}}\big[\overleftarrow{\mu}_{t+1,\texttt{nx}_{ti}} \cdot \phi_{ti}\big] \tag{5.8}$$

After computing both sets of messages, all information about each node will be contained in the local posterior beliefs

$$\mathcal{B}_{ti}(x_{ti}^{\texttt{pr}}, x_{ti}) = \overrightarrow{\mu}_{t-1,\texttt{pr}_{ti}} \cdot \phi_{ti} \cdot \overleftarrow{\mu}_{t+1,\texttt{nx}_{ti}} \tag{5.9}$$

Note that $\mathcal{B}_{ti}$ is proportional to the marginal posterior over $(x_{ti}, x_{ti}^{\texttt{pr}})$, but does not necessarily integrate to one.

At this point, we have computed the posterior over X by assuming a fixed value of L. To optimize over L we consider the marginal likelihood of a given track, computed by integrating out all X variables. This quantity can be efficiently retrieved from any node along the track as

$$\mathcal{M}_{ti} = \int \mathcal{B}_{ti}(x_{ti}^{\texttt{pr}}, x_{ti}) \, \mathrm{d}x_{ti}^{\texttt{pr}} \, \mathrm{d}x_{ti} \tag{5.10}$$

Eq. 5.10 allows us to maximize the marginal likelihood of all tracks present at t over L^t while holding $L^{(t')}$ fixed for $t' \neq t$:

$$L^{*(t)} = \underset{L^{(t)}}{\operatorname{argmax}}\, \mathbf{P}(Z \mid L) = \underset{L^{(t)}}{\operatorname{argmax}} \prod_i \mathcal{M}_{ti} \tag{5.11}$$

This optimization can be solved as a Linear Assignment Problem (LAP) between the nodes at t and $t-1$ formulated using the binary indicator matrix $L^{(t)}$:

$$\phi_{tij} \equiv \mathbf{P}(z_{ti} \mid x_{ti}) \cdot \mathbf{P}\Big(x_{ti} \,\Big|\, x_{t-1,j}, L^{(t)}_{ij}{=}1\Big) \tag{5.12}$$

$$\mathcal{M}_{tij} := \int \overrightarrow{\mu}_{t-1,j} \cdot \phi_{tij} \cdot \overleftarrow{\mu}_{t+1,\mathbf{nx}_{ti}}\, \mathrm{d}x_{t-1,j}\, \mathrm{d}x_{ti} \tag{5.13}$$

$$L^{*(t)} := \max_{L^{(t)}} \sum_{ij} L^{(t)}_{ij} \cdot \log \mathcal{M}_{tij} \tag{5.14}$$

Note that $\mathcal{M}_{tij}$ is the hypothetical value of $\mathcal{M}_{ti}$ if we had torn the node (t,i) from its current assignment and attached it to node $(t-1,j)$ instead.

Picking a new value of $L^{(t)}$ according to Eq. 5.14 is the local model selection step of the algorithm. Doing so, we are picking a consistent subset of the potentials $\{\phi_{tij}\}_{ij}$ which is maximally compatible with the incoming messages. As shown in Chapter 4, this operation does not affect any of the forward messages before time t or any of the backward messages after time t — these only depend on values of $L^{(t')}$ for $t' < t$ and $t' > t$ respectively. After updating the value of $L^{(t)}$, we continue message passing in the forward direction using the selected set of potentials. This allows us to interleave optimization over $L^{(t)}$ into the standard message passing procedure. We use the messages $\{\overrightarrow{\mu}_{t-1}\}$ and $\{\overleftarrow{\mu}_{t+1}\}$ to update $L^{(t)}$, and subsequently use the new value of $L^{(t)}$ to compute the forward messages $\{\overrightarrow{\mu}_{t}\}$. Virtual nodes with no observations are added at time t for any nodes from $t-1$ which were left unassigned. The process is repeated going forward; at each point increasing the marginal likelihood $\mathbf{P}(Z|L)$. For the sake of simplicity, the backward pass of the algorithm remains unchanged from a standard smoother and does not modify the assignments. This ILMS forward-backward procedure is repeated until convergence. An outline of the modified forward pass is listed in Fig. 5.3.

procedure FORWARDMESSAGEPASS
  for $t = 1 \dots T$ **do**
    remove all virtual nodes at t
    for all $n = (t, i)$, $n' = (t-1, j)$ **do**
      compute $\mathcal{M}_{tij}$ using Eq. 5.13
    end for
    re-estimate $L^{(t)}$ using Eq. 5.14
    add virtual nodes at t
    for all $n = (t, i)$ **do**
      update forward message $\overrightarrow{\mu_{ti}}$ using Eq. 5.7
    end for
  end for
end procedure

Figure 5.3: Approximate message passing procedure used for inference in the forward direction. Virtual nodes are added to extend tracks which have been determined to have no 'non-virtual' continuations.

5.6 Tracking by Detection with Latent Data Association

Up to this point we have described the Latent Data Association parameterization and inference algorithm in general terms. We now describe the implementation details and extensions used for the presented evaluations. To this end we describe the observation and state space models for both 2D and 3D tracking, as well as extensions to handle false positive detections and track length priors. Fig. 5.4 illustrates the graphical model for a single node with the modifications described in this section.

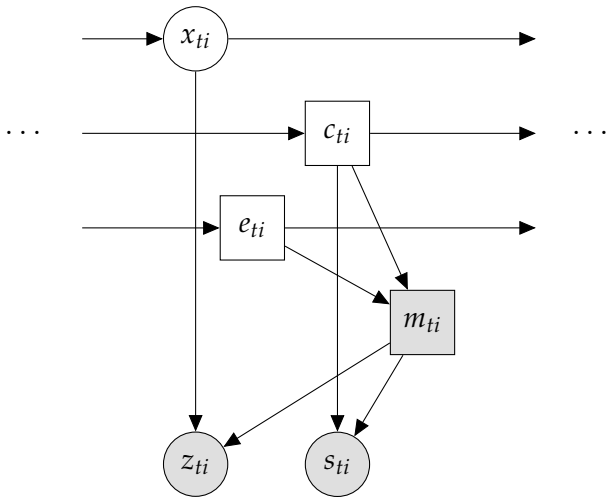


Figure 5.4: Extended model for a single node $n = (t, i)$ with extra variables to account for the track category, detector score, missing observations, and track termination.

Since every detection now corresponds to a track, outliers must correspond to outlier tracks, leading to an extra discrete state variable, $c_{ti} \in \{\texttt{pedestrian}, \texttt{outlier}\}$, representing the target class. To go with the class model, a prior $\mathbf{P}(c_{ti})$ and transition model $\mathbf{P}\left(c_{ti} \,\middle|\, c_{t-1,j}, L_{ij}^{(t)}=1\right)$ must be defined. In our evaluation, we use only two classes, but in principle the formulation allows for more.

The pedestrian detectors we use are discriminative, so no generative model exists to explain the observations based on the target class. To compensate, we train the observation model for the detector. The score of each detector firing is treated as a real-valued observation, s_{ti}, conditioned on the class. Kernel Density Estimation (Gaussian kernel with a width of 0.05) is used to estimate the distributions $\mathbf{P}(s_{ti} \mid c_{ti})$. The distribution is trained by matching detector firings with ground truth annotations over sequences out of the PETS'09 dataset [155] (the `S2.L1` sequence is excluded since it is used for evaluation). Fig. 5.5 shows the conditional distributions of the trained model.

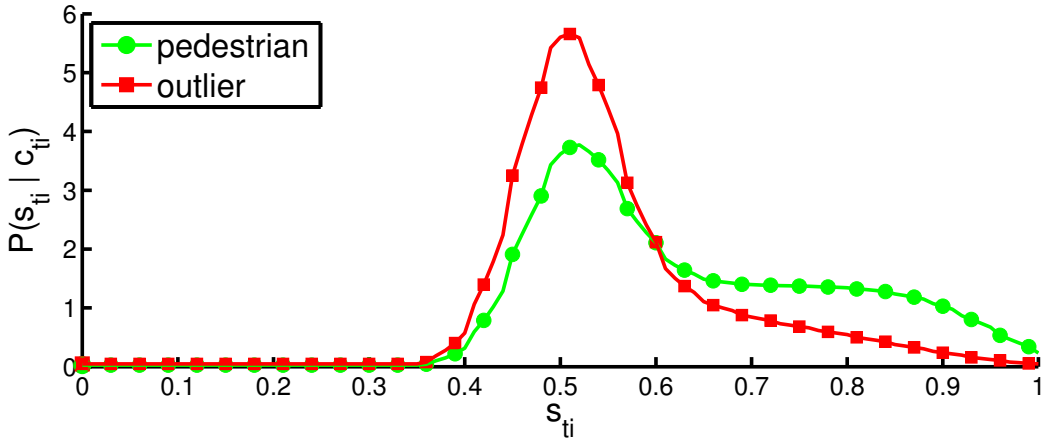


Figure 5.5: Learned model of the object detector firing score s_{ti} conditioned on the object class $c_{ti} \in \{\texttt{pedestrian}, \texttt{outlier}\}$.

In practice, a lot of information is contained in the missing detections – a track with very few detections is more likely to be an outlier than one with many consistent detections. To incorporate this negative information, we include detector failure into the observation model. The indicator variable $m_{ti} = 1$ is used to denote a missing observation at node $n =$

(t, i). In this case n is a virtual node and the z_{ti} and s_{ti} observation variables are ignored. We allow missing observations to occur with probability dependent on the underlying class.

Finally, we include a track length prior. Because of the detector failure model, we cannot assume a track continues on indefinitely after its last observation – doing so would imply a very large number of missing observations and make all tracks likely to be outliers. Instead, we give each target track a fixed probability of terminating at every time instance after its last observation. We introduce the indicator variable e_{ti} to mark that the track has ended. Once this variable transitions from 0 to 1, a transition in the other direction is not possible. If $e_{ti} = 1$, we require that $m_{ti} = 1$. This encodes the fact that once a track ends it cannot be observed.

5.7 Modified Inference Procedure

Incorporating the changes of Section 5.6 into the approximate inference procedure described in Section 5.5 is not difficult since all of the modifications can be represented as additional discrete components in the Markov chain. Furthermore, Eq. 5.9 and Eq. 5.10 do not depend on the Markov chain being continuous; analogous equations hold for a discrete chain if the marginalization integrals are replaced with sums. We run discrete message passing over e_{ti} and c_{ti} and compute the combined track score by adding the log-likelihoods obtained from Eq. 5.10 applied to the discrete and continuous Markov chains independently. As before, we update $L^{(t)}$ by solving the LAP in Eq. 5.14 with the cost of each assignment now based on the combined track log-likelihood from both the discrete and continuous chain. The discrete and continuous chains are kept separate because we are using a linear assignment problem to pick $L^{(t)}$. This prevents modeling dependencies between c_{ti} and x_{ti}, but allows for a more efficient algorithm. Modifying the local optimization procedure to allow such dependencies is a possible target for future research.

5.8 Evaluation

Experimental validation was performed using four publicly available video sequences comprising over 1200 frames from two standard pedestrian tracking datasets (TUD [122] and PETS'09 [155]). 2D tracking was used for the TUD datasets and 3D tracking for the PETS sequence. We ran 2D tracking on `TUD-Stadtmitte` despite the available camera calibration because the oblique viewing angle makes accurate estimation of ground plane positions difficult. Raw detections, ground truth annotations, and tracking area specifications provided by Andriyenko *et al.* [139] were used for all evaluations. Results are presented in terms of the CLEAR MOT[156] metrics for tracking performance and precision-recall curves for classification accuracy. We also include the number of fragmentations (FM), mostly tracked targets (MT), and identity switches (IDS). All evaluations use a 50% intersection over union threshold for matching 2D bounding boxes.

A constant-velocity motion model with direct linear observations was used within each track:

$$x_0 \sim \mathcal{N}(\mu_0, \Sigma_0) \tag{5.15}$$

$$x_{t+1} \sim \mathcal{N}(A \cdot x_t, \Sigma_{\texttt{mot}}) \tag{5.16}$$

$$z_t \sim \mathcal{N}(B \cdot x_t, \Sigma_{\texttt{obs}}) \tag{5.17}$$

In the above, A implements the constant-velocity model and the B selects the bounding box position and dimensions out of the state space.

In the 2D case, the continuous state space is composed of the bounding box center and the log of the dimensions. Dimensions are tracked in log-space to help compensate for perspective effects. Both the position, $\mathbf{p}$, and log-dimensions, $\mathbf{d}$, have an associated velocity ($\dot{\mathbf{p}}$ and $\dot{\mathbf{d}}$) resulting in an 8D state space: $(\mathbf{p}_x, \mathbf{p}_y, \mathbf{d}_x, \mathbf{d}_y, \dot{\mathbf{p}}_x, \dot{\mathbf{p}}_y, \dot{\mathbf{d}}_x, \dot{\mathbf{d}}_y)$. The position prior is centered in the image with mean log-dimensions of $\log(320)$ by $\log(240)$. The standard deviation (s.d.) is 400px for the position and 1.0 for the log-dimensions. We incorporate a correlation coefficient of 0.99 between the prior log-dimensions. The velocity prior is

zero-mean with an s.d. of 5px for the center location and 0.01 for the log-dimensions. The motion model adds isotropic noise with an s.d. of 10^{-4}px, 10^{-4}, 0.5px/s, and 10^{-2}, for the $\mathbf{p}$, $\mathbf{d}$, $\dot{\mathbf{p}}$, and $\dot{\mathbf{d}}$ components respectively. The observation model is unbiased with an s.d. of 10px for $\mathbf{p}$ and 0.1 for $\mathbf{d}$.

For 3D tracking, object position is tracked on the ground plane together with the bounding box dimensions (width and height are tracked; depth is assumed equal to width). We again use a constant-velocity model for the ground plane position, but assume the dimensions follow a random walk with no velocity (unlike in the 2D tracking case, we expect the 3D dimensions to stay relatively constant). The 3D state space consists of $(\mathbf{p}_x, \mathbf{p}_y, \mathbf{d}_x, \mathbf{d}_y, \dot{\mathbf{p}}_x, \dot{\mathbf{p}}_y)$. The prior is zero mean for $\mathbf{p}$ and $\dot{\mathbf{p}}$ with an s.d. of 40m and 0.25m/s respectively. The prior for $(\mathbf{d}_x, \mathbf{d}_y)$ has mean $(0.7\text{m}, 1.7\text{m})$ with an s.d. of 0.2m. The constant velocity motion model adds isotropic noise with an s.d. of 10^{-4}m, 0.05m/s, and 0.01m for the three components of the state space respectively. We assume observation noise with an s.d. of 0.15m for the position and 0.20m for the dimensions.

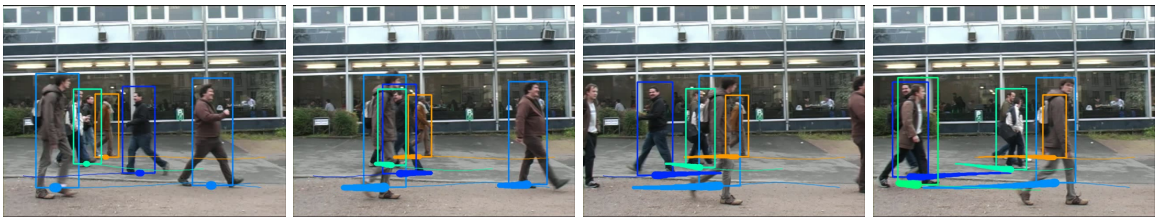
The discrete model parameters are the same for both 2D and 3D tracking. We use a uniform prior over $\mathbf{P}(c_0)$, and a transition model such that $\mathbf{P}(c_t = c_{t-1}) = 1 - 10^{-6}$. The missing detections probability, $\mathbf{P}(m_t \mid e_t = 0, c_t)$, is 0.6 for pedestrians and 0.7 for outliers. The track termination probability, $\mathbf{P}(e_t = 1 \mid e_{t-1} = 0, c_t)$, is set so there is a 0.0025(pedestrian) and 0.18(outlier) chance of terminating after one second. These parameters were determined empirically and scaled based on the frame rate $\frac{1}{\Delta t}$ where appropriate. We note in particular that the discrete Markov transition matrix, T, is adjusted to $T^{\Delta t}$ to control for the frame rate. Because our system keeps track of object size as well as location, the size of the bounding boxes output by the detector vs the size of the labeled ground truth plays an important role in the performance of the system. We scale the width of the bounding boxes output by our system by 0.75 to better match the ground truth labeling.

The output of our tracking algorithm on the evaluation sequences in shown in Fig. 5.6 as a series of filmstrips. Quantitative results are shown in Tab. 5.1 and are competitive

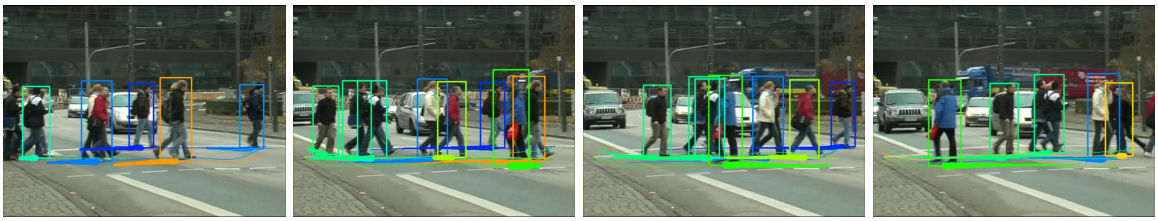
with the state of the art. We note that despite the widespread use of the CLEAR MOT metrics, direct comparison of published algorithms is still difficult as many authors differ in the precise evaluation methods used (2D v.s. 3D metrics, different regions of interest, etc.). Despite this, we have attempted to make an informative evaluation against recently published results – we do not imply a head-to-head comparison. Where 2D evaluations are available, we list those published by the authors. To compare with Andriyenko *et al.* [139], we have run our own 2D evaluation scripts on the their data where possible, as well as listing their published results. Only 3D ground tracks were available for the `TUD-Stadtmitte` sequence. In this case we assumed average 3D pedestrian dimensions and projected these into 2D bounding boxes.

Fig. 5.7 shows the marginal log-likelihood as a function of the number of forward-backward iterations. Note the monotonic increase in log-likelihood and convergence in a small number of iterations.

Precision Recall curves showing improvement over the baseline detector are show in Fig. 5.8. These curves are possible because of the probabilistic nature of our approach where each output has an associated posterior pedestrian vs outlier probability.

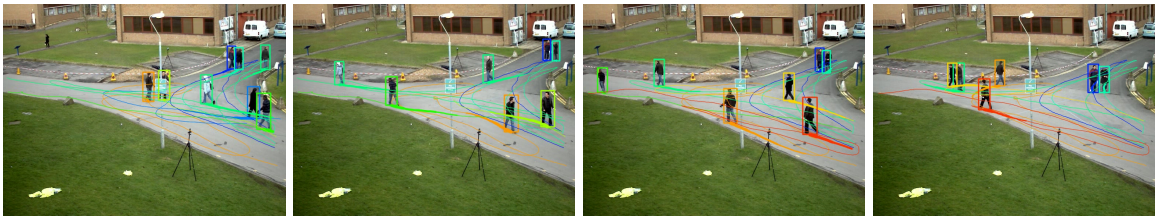


(a) TUD-Campus



(b) TUD-Crossing

(c) TUD-Stadtmitte



(d) PETS'09 S2.L1 (View 1)

Figure 5.6: Filmstrip sequences showing tracking output on the evaluation sequences. Outlier tracks are not shown.

Algorithm		MOTA	MOTP	IDS	MT	FM
proposed		0.82	0.74	0	5	3
Breitenstein2011[4]	[150]	0.67	0.73	2	–	–

(a) TUD-Campus

Algorithm		MOTA	MOTP	IDS	MT	FM
proposed		0.74	0.76	2	7	12
Zamir2012[4]	[157]	0.92	0.76	0	–	–
Breitenstein2011[4]	[150]	0.71	0.84	2	–	–

(b) TUD-Crossing

Algorithm		MOTA	MOTP	IDS	MT	FM
proposed		0.73	0.71	2	4	1
Zamir2012[4]	[157]	0.78	0.63	0	–	–
proposed[2]		0.63	0.73	4	4	1
Andriyenko2012[2,3]	[139]	0.61	0.68	3	6	1

(c) TUD-Stadtmitte

Algorithm		MOTA	MOTP	IDS	MT	FM
proposed		0.90	0.75	6	17	21
Zamir2012[4]	[157]	0.90	0.69	8	–	–
Andriyenko2012[3]	[139]	0.79	0.66	29	17	56
Andriyenko2012[1,4]	[139]	0.89	0.56	–	–	–
Breitenstein2011[1,4]	[150]	0.56	0.80	–	–	–
proposed[2]		0.92	0.75	4	18	18
Andriyenko2012[2,3]	[139]	0.83	0.65	24	18	43

(d) PETS'09 S2.L1 (View 1)

[1] evaluated by PETS'09 workshop
[2] cropped to tracking region of Andriyenko *et al.* [139, 138]
[3] our own 2D evaluations using authors' provided output data
[4] results as published by authors

Table 5.1: A comparison using various tracking metrics. The threshold $\mathbf{P}(c_{ti} = \texttt{pedestrian}) \geq 0.50$ was used for all evaluations of our algorithm. Note that Zamir *et al.* [157] makes use of appearance information, so better performance is expected.

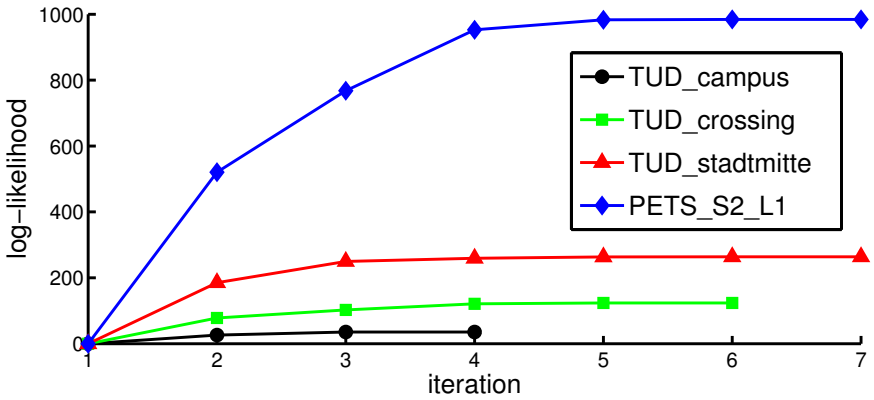
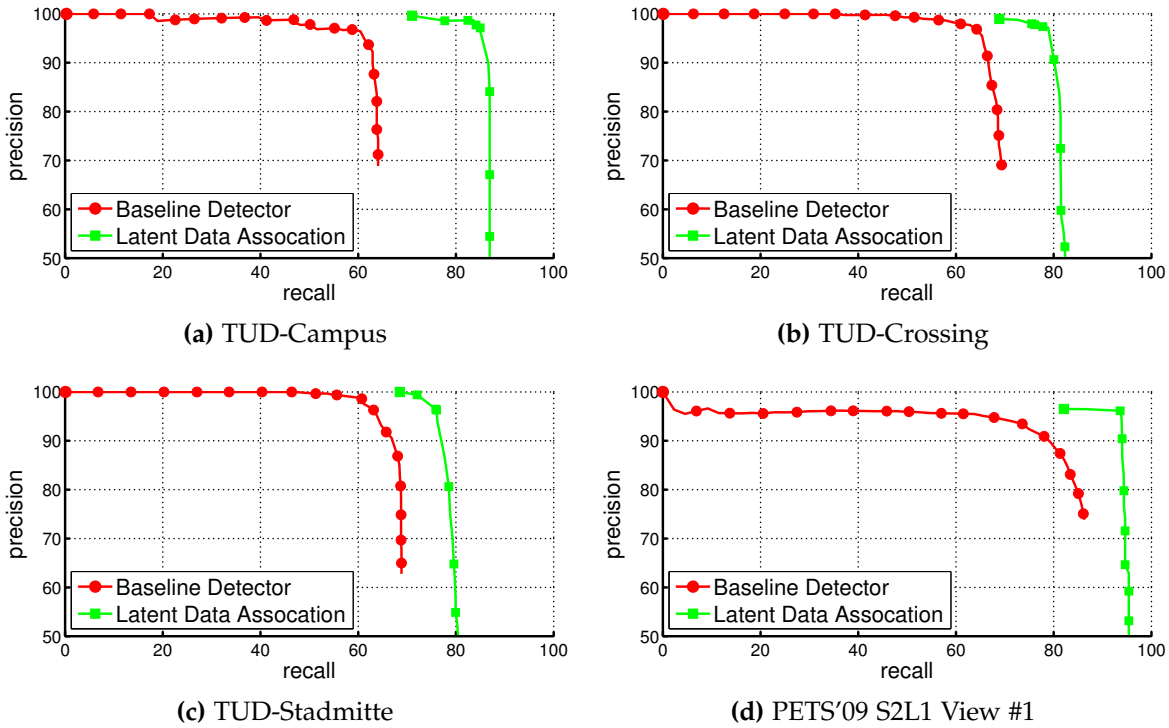


Figure 5.7: Convergence of the approximate inference algorithm is achieved in under 7 iterations for all evaluated sequences. Each curve has been zeroed to its initial log-likelihood.



(a) TUD-Campus **(b)** TUD-Crossing

(c) TUD-Stadmitte **(d)** PETS'09 S2L1 View #1

Figure 5.8: Precision-Recall curves for all datasets plotted alongside the baseline detector.

5.9 Conclusions

In this chapter we have presented a novel multi-target tracking algorithm based on the Iterative Local Model Selection framework. In order to apply ILMS to the problem, we treat data association variables as the 'switches' in a Switching Linear Dynamical Systems. A key component of the proposed algorithm is the use of a 'latent' parameterization of the data association problem where associations are made between the latent state variables. The major advantage of this parameterization is that it implicitly determines the number of targets in the environment, avoiding various heuristics commonly used to make this determination. Having compressed the entire inference problem into an SLDS, we add additional discrete variables to model a few properties of the targets. These include an inlier vs outlier classification and a rudimentary track length prior. The combined system has been compared against various state-of-the-art methods and shown to be competitive in terms of performance. We note that the qualitatively, the tracks output by the algorithm are very smooth and natural, without any glaring inconsistencies. Given that these results were obtained without any use of an appearance model, attempting to add some measure of visual similarity into the track matching criteria could further improve results.

Acknowledgment

This work was supported by the Engineering and Physical Science Research Council [grant number EP/H050795] and the Australian Research Council, grant DP130104413.

6

Robust Pose Graph Estimation

6.1 Introduction

This chapter demonstrates how the Iterative Local Model Selection algorithm can be applied to large-scale robust mapping. Robustness is achieved by including discrete indicator variables which are used to model outliers, resulting in a robust estimation algorithm. Unlike the tracking application in the previous chapter, pose graph estimation cannot be modeled as a Switching Linear Dynamical System (SLDS) and requires a more general Conditional Linear Gaussian Network (CLGN) model. On top of this it is fundamentally a non-linear optimization problem rather than a linear inference problem. This chapter demonstrates two adaptations to ILMS which make it applicable to this scenario. First, we describe how ILMS can be made to work on CLGNs using a specific message passing order. Second, we demonstrate that it can be combined with the Gauss-Newton algorithm and applied to non-linear estimation problems. Fig. 6.1 shows an example of the results generated by the system described in this chapter.

Most recent work on SLAM for large scale environments has adopted the pose graph formulation combined with non-linear least squares as the preferred approach. Due to its simplicity, the technique is able to scale to extremely large datasets and has been used

to build systems capable of operating over hundreds of kilometers[158]. These systems typically rely on external loop closure detection algorithms in order to generate constraints between non-consecutive poses in the graph. Popular approaches such as FabMap[75] and that of Cadena et al. [159] use visual features to recognize when the robot revisits a previously seen location.

Unfortunately, detecting loop closures based on visual information is a difficult problem with incorrect detections always a possibility due to perceptual aliasing, among other sources of error. Even state-of-the-art loop closure detection algorithms are not perfect, and as the size of the map increases incorrect loop closures will inevitably make it past the front end into the back end of the SLAM system.

The non-linear least squares pose graph formulation is extremely sensitive to such incorrect constraints being allowed into the pose graph. While robust estimators can be used, these are not sufficiently robust to deal with all generated candidates and require conservative filtering of the constraints. This has motivated research on robust pose graph optimization algorithms which go beyond classical robust estimation techniques and deal with incorrect loop closures directly in the back end.

In a broader sense, long-term SLAM systems will inevitably confront situations where the correct map depends on a discrete set of possible explanations for what was observed by the sensors. It is therefore essential that the back-end optimization must be able to handle multimodality. In addition to loop closures, such situations could be caused by ambiguous odometry (e.g. wheel slip[160]) or a changing environment. For these reasons, dealing with discrete choices in the SLAM optimization framework is a key problem for long-term autonomous mapping and navigation.

We propose an approach based on interpreting the linearization of the switched non-linear optimization objective as a Conditional Linear Gaussian Network (CLGN). We use the ILMS strategy from Chapter 4 as an approximate inference algorithm in order to estimate subsequent linearization points. In this way our algorithm generates a sequence of

approximate solutions which locally minimizes the linearized objective in terms of both the discrete and continuous variables.

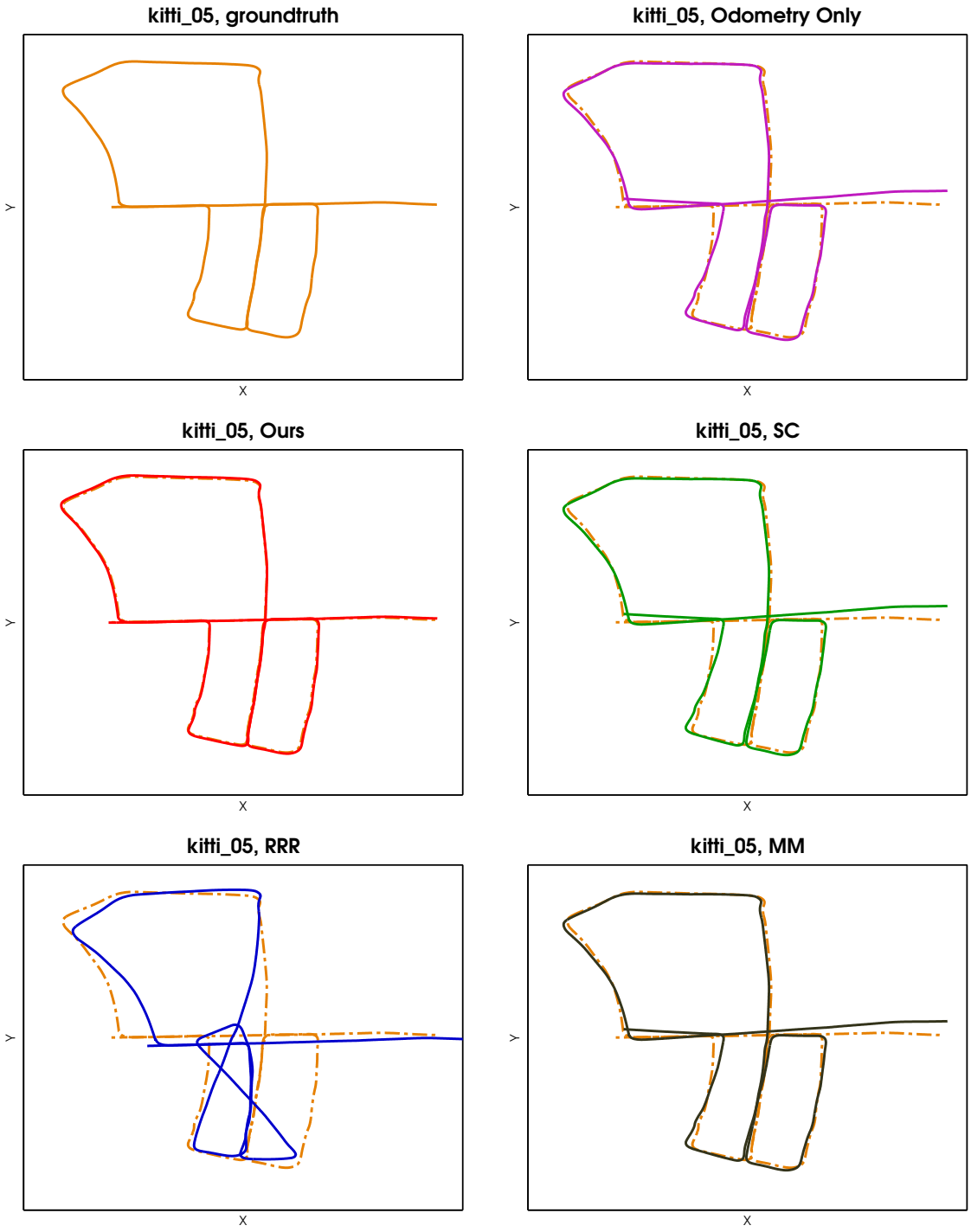


Figure 6.1: The trajectory output of the algorithms we evaluated alongside groundtruth (dashed/orange) and raw odometry tracks for one of the harder test sequences. Our approach correctly identifies the loop closures and produces an accurate track.

6.2 Related Work

Formulations of Simultaneous Localization and Mapping (SLAM) as global optimization over a constraint network have a long history in the literature[40]. One of the first modern algorithms was introduced by Lu and Milios [49]. *GraphSLAM*[161] further popularized and refined the technique by improving numerical stability through the use of information rather than covariance matrices. More recently, *g2o*[83] has provided a general software framework for algorithms formulated as sparse constraint graphs. While *g2o* offers the option of robust kernels[82], these methods are not insufficiently robust when large amounts of outlier loop closures are present[162].

As an alternative to *g2o*, work by Kaess *at al*[72] on the *iSAM* and *iSAM2* algorithms uses inference and message passing instead of the traditional sparse linear algebra. The key idea is a connection between the internals of sparse matrix factorization and inference on graphical models. We draw inspiration from this, but while *iSAM2* does not deal with discrete variables, we extend the methodology to do so. We note that unlike our work, *iSAM2* uses a "square root" form of message passing which corresponds to QR decomposition of the Jacobian matrix. Although this has advantages in terms of numerical stability, it is not a principled difference in the context of this chapter. A more significant difference is that *iSAM2* does not explicitly compute the local posterior information matrices. Combined with the traditional, two phase message passing order assumed by the architecture, this makes it difficult to implement our algorithm in the GTSAM library.

Pinies et al. [163] also propose a Junction Tree algorithm for sparse bundle adjustment. This work is more similar to ours in that it is based on a Cholesky decomposition of the information matrix. More importantly, the algorithm computes the local posterior information matrices for each clique in the Junction Tree. Our approach goes further by adding discrete variables to the cliques and optimizing them during the course of the message passing.

Recently, a series of three publications on robust pose graph estimation has targeted the same problem. Of these, the *Max-Mixture* (MM)[160] and *Switchable Constraints* (SC)[81] algorithms are general purpose techniques, while the *Realizing, Reversing, Recovering* (RRR)[164] approach is tailored specifically to robust pose graph estimation.

RRR creates clusters of loop closure detections based on proximity in the topography of the pose graph. Starting with just the odometry map, the algorithm attempts to add these clusters into the map while maintaining consistency with the odometry data. A full minimization of the error is carried out in order to test the compatibility of a cluster to the odometry, at which point a threshold is applied to decide if it should be kept.

The *Switchable Constraints*[81] algorithm is more general than RRR, but is only capable of *removing* constraints from the graph – e.g. it cannot decide between mutually exclusive options. A weight variable is added into the nonlinear objective for each switchable constraint. The objective is multiplied by this weight in order to create a continuous relaxation of the full discrete-continuous mixture. The set of weight are optimized together with the rest of the objective and are given a prior pulling them towards 1. Agarwal et al. [165] propose the *Dynamic Covariance Scaling* algorithm which computes a closed form approximation to each of the SC weight variables and thus avoids increasing the size of the optimization problem. This modification is similar in spirit to Iteratively Reweighted Least Squares (IRLS) and shows improved performance over the standard form of SC in both accuracy and speed. Along similar lines, Lee *at al*[166] propose an Expectation-Maximization (EM) based derivation of an IRLS-like algorithm with a Cauchy weighting kernel. They show this approach to be more robust than the typical Huber M-estimator[82].

Max-Mixtures is the most general of the three basic approaches, allowing arbitrary mixtures of objectives to be placed into standard nonlinear optimization. The key is to replace the sum in the likelihood of a mixture model with a maximum. When computing the log-likelihood of such a mixture, the sum cannot be pulled out of the logarithm whereas the max can. This converts objectives of the form $\log(\phi_1(x) + \phi_2(x))$ into

$\max\{\log(\phi_1(x)), \log(\phi_2(x))\}$. The modified objective can be optimized by a standard non-linear optimizer such as *g2o*.

Our approach is as general as *Max-Mixtures*, but offers better performance than all of the evaluated methods on the robust pose graph estimation problem. Fig. 6.1 and Fig. 6.8 shows the output tracks for two of the evaluation sequences. In the following section we cover the background and notation needed to introduce our algorithm in Section 6.6.

6.3 Pose Graphs

The pose graph optimization problem is well known in the robotics community. The goal is to reconstruct the path of a robot traversing an unknown environment. A sequence of robot poses, $p_t = (x_t, y_t, \psi_t)$, is given with constraints between pairs of poses based on odometry information as well as loop closures. A single constraint between a pair of poses indexed by s and t is of the form

$$g_{s,t}(p_s, p_t) = \frac{1}{2}||(p_s \oplus m_{s,t}) \ominus p_t||^2_{\Sigma_{s,t}} \tag{6.1}$$

where $\oplus$ and $\ominus$ are the standard pose composition operators[38], $m_{s,t}$ is a measurement of the relative transformation between the two poses, and $\Sigma_{s,t}$ is the covariance matrix for the measurement.

The constraints form a network of measurements relating the various poses. Consecutive poses are always related by an odometry measurement. Occasionally, an external loop closure detection algorithm[167][168] reports that the robot has returned to a location it has previously seen. In this case, a constraint is created between the current pose and a previous pose. Because loop closure detections are imperfect, they must be conservatively filtered so as not to corrupt the resulting constraint network with spurious, incorrect constraints.

Taken together, the pose graph optimization problem can be solved by minimizing the

non-linear objective

$$\sum_t g_{t,t+1} + \sum_{(s,t)\in L} g_{s,t} \tag{6.2}$$

where L is the set of loop closure constraints. After ensuring L is free from incorrect loop detections, Eq. 6.2 is typically optimized with a nonlinear least squares solver such as *g2o*[83] or *iSAM2*[72]. This optimization will fail if false-positive loop closures are not removed.

6.4 Nonlinear Least Squares

Nonlinear estimation is typically performed with a variant of the Gauss-Newton algorithm. Given a set of functions $f_i(x_i)$ (with each x_i a subset of the problem domain X) and the associated covariances Σ_i, the goal is to find a value for all variables which minimizes

$$\sum_i \frac{1}{2} f_i(x_i)^\top \cdot \Sigma_i^{-1} \cdot f_i(x_i) \tag{6.3}$$

The Gauss-Newton algorithm finds such a point by approximating each function $f_i(x_i)$ with its linearization at x_i^*

$$\tilde{f}_i(x_i) = f_i(x_i^*) + \nabla f_i(x_i^*) \cdot x_i \tag{6.4}$$

With this approximation in place, Eq. 6.3 becomes a linear least squares problem which can be solved exactly

$$\underset{x}{\operatorname{argmin}} \sum_i \frac{1}{2} \tilde{f}_i(x_t)^\top \cdot \Sigma_i^{-1} \cdot \tilde{f}_i(x_t) \tag{6.5}$$

The Gauss-Newton algorithm proceeds by taking the solution of Eq. 6.5 as the new linearization point, and iteratively repeating the process until convergence.

6.5 Relinearization as Inference

A key aspect of the Gauss-Newton algorithm is solving Eq. 6.5 to obtain a new linearization point which is likely to decrease the value of Eq. 6.3. This process can be thought of as an inference procedure based on the correspondence between least squares and the Gaussian distribution.

We now review this correspondence and define notation for the rest of the chapter. Consider a Gaussian distribution over z with mean $\mu(x)$

$$\begin{aligned} \mathbf{P}(z \mid x) &= \mathcal{N}(z\,;\, \mu(x), \Sigma) \\ \mu(x) &= B \cdot x \end{aligned} \tag{6.6}$$

If z is observed, the maximum likelihood solution for x corresponds to solving the least squares problem

$$\underset{x}{\operatorname{argmin}} \frac{1}{2}||B \cdot x - z||^2_{\Sigma} \tag{6.7}$$

Distributions of the sort defined by Eq. 6.6 are Gaussian potentials as defined in Chapter 3. The log-likelihood of a Gaussian potential is defined via the associated quadratic form

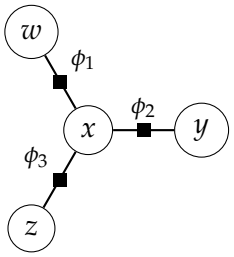
$$\log \phi(x, z) \equiv \frac{1}{2}(x, z)^{\top} \cdot A \cdot (x, z) + (x, z)^{\top} \cdot b + c \tag{6.8}$$

parameterized by the matrix A, vector b, and constant c. Since the linearized objectives in Eq. 6.5 are also of this form, we can interpret each term as a Gaussian potential. Assuming the residual $f_i(x_i) \in \mathbb{R}^k$,

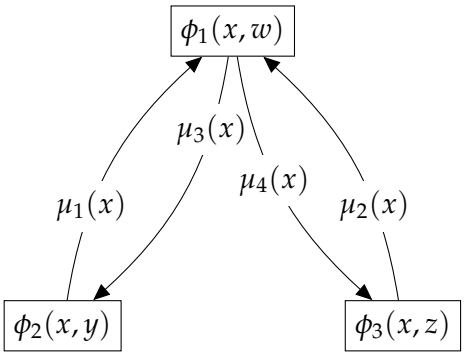
$$\log \phi_i(x_i) \equiv \frac{1}{2} x_i^{\top} \cdot A_i \cdot x_i + x_i^{\top} \cdot b_i + c_i \equiv -\frac{1}{2}\tilde{f}_i(x_i)^{\top} \cdot \Sigma_i^{-1} \cdot \tilde{f}_i(x_i) - \frac{1}{2}|\Sigma_i| - \frac{k}{2}\log(2\pi) \tag{6.9}$$

Thus, the network of linearized constraints in the pose graph can be thought of as a Gaussian Random Field, and minimizing Eq. 6.5 is equivalent to maximizing the likelihood of this network.

Maximum likelihood on Gaussian Random Fields is a well studied topic[95]. One standard solution to the problem is based on converting the factor graph into a clique tree on which efficient inference can be performed. This conversion is analogous to picking a variable ordering in sparse matrix solvers (i.e. COLAMD[169]) and was described in Section 3.7. Fig. 6.2a shows a sample factor graph over a small network of three Gaussian potentials. Fig. 6.2b illustrates a clique tree created from this factor graph. Once such a tree is created, it can be used to compute the maximum likelihood values of the variables efficiently using the Belief Propagation (BP) algorithm[170].



(a) A factor graph composed of three purely continuous Gaussian potentials each representing a linearized constraint.



(b) Standard Belief Propagation on the associated clique tree. Messages are indexed based on the order in which they are computed.

Figure 6.2: An example of a purely continuous Gaussian model and a run of the standard Belief Propagation algorithm on the corresponding clique tree.

The BP algorithm uses dynamic programming in the form of message passing to compute the full maximum-likelihood state by computing partial maximizations of the network one piece at a time. Each message performs one such maximization, and the result is cached for later re-use. This allows the algorithm to compute the maximum-likelihood estimates for the whole constraint network without ever creating the full product distribution over all variables. Eq. 6.10 shows the messages computed in a run of the BP algorithm on the clique tree in Fig. 6.2b.

$$\begin{aligned} \mu_1(x) &= \max_y \phi_2(x,y) \quad & \mu_3(x) &= \max_w \phi_1(x,w) \cdot \mu_2(x) \\ \mu_2(x) &= \max_z \phi_3(x,z) \quad & \mu_4(x) &= \max_w \phi_1(x,w) \cdot \mu_1(x) \end{aligned} \tag{6.10}$$

Once the messages are computed, we can retrieve the optimal value of (x, w), for example, by computing

$$(x^*, w^*) = \operatorname*{argmax}_{x,w} \phi_1(x,w) \cdot \mu_1(x) \cdot \mu_2(x) \tag{6.11}$$

The messages μ_1 and μ_2 capture information from the rest of the constraints and allow us to locally compute an optimal value for (x^*, w^*) which takes into account global information. For a general clique tree, message passing proceeds in two phases. In the first phase, messages are passed upward towards the root of the tree, collecting information from all of the nodes in one place. In the second phase, messages are passed downward from the

root to the children. After both sets of messages are computed, all constraint information has been propagated throughout the network.

6.6 Hybrid Inference Optimization

The relationship between Least Squares and Gaussian Graphical Models outlined in the previous section suggests that alternative inference algorithms could be used to predict the next linearization point; this idea forms the basis of our algorithm. Our approach replaces the continuous inference procedure used by Kaess et al. [72] with the Iterative Local Model Selection procedure described in Chapter 4. The result is a nonlinear optimization algorithm capable of dealing with mixture model objectives.

In the application to pose graph estimation, incorrect loop closures are dealt with by adding $d_{s,t} \in \{0, 1\}$ as a discrete variable for each loop closure $(s, t) \in L$. When $d_{s,t} = 1$, the original constraint $g_{s,t}$ is active, whereas $d_{s,t} = 0$ means an alternative, outlier objective $g_{s,t}^{\texttt{out}}$ is used in its place. The outlier objective is simply Gaussian with a very large covariance matrix, identical to the outlier objectives used by Olson and Agarwal [160]. The algorithm used to optimize this objective, however, is different.

In order to introduce our algorithm we need to incorporate discrete switch variables into the previously defined notation. The objectives of Eq. 6.3 will now take the form $f_i(x_i, d_i)$ with each d_i a single discrete variable:

$$\begin{aligned} \min . \quad & \sum_i \frac{1}{2} f_i(x_i, d_i)^\top \cdot f_i(x_i, d_i) \\ & d_i \in \{0, 1\} \end{aligned} \tag{6.12}$$

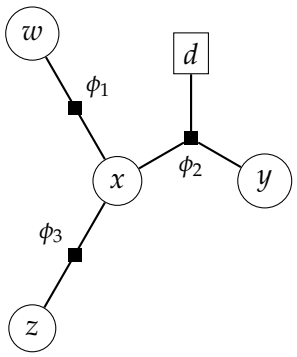
We use i to index the full set of objectives/constraints with each f_i corresponding to a $g_{s,t}$ and d_i to the matching $d_{s,t}$. This is done to minimize clutter in the notation.

With a switching objective function, linearizing f_i no longer results in a single Gaussian potential. Instead, we get a hybrid Gaussian potential as defined Section 3.8. The hybrid potential consists of a table of Gaussian potentials with one entry for each value of d_i. Each

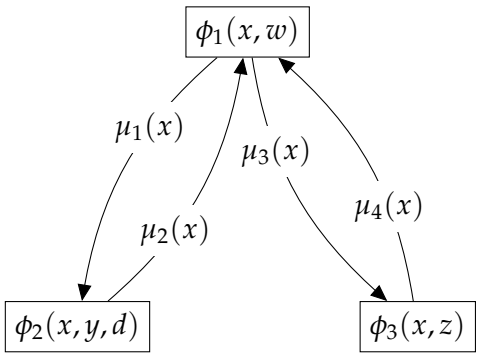
of these is in turn parameterized by a quadratic form.

$$\log \phi_i(x_i, d_i{=}k) \equiv \log \phi_i^{(k)}(x_i) \equiv \frac{1}{2} x_i^\top \cdot A_i^{(k)} \cdot x_i + x_i^\top \cdot b_i^{(k)} + c_i^{(k)} \tag{6.13}$$

Fig. 6.3a shows the earlier mentioned factor graph modified to include a discrete variable, d. Fig. 6.3b shows the associated clique tree, which we will use to illustrate the necessary modifications to the basic ILMS algorithm. Note that the message passing order has changed, as will be explained in Section 6.6.1.



(a) A hybrid factor graph composed of three potentials. The ϕ_2 potential is hybrid since it contains the discrete variable d in its domain.



(b) The corresponding clique tree showing the message passing order required to ensure convergence of ILMS. Messages are indexed based on the order in which they are computed.

Figure 6.3: A sample Conditional Linear Gaussian Network with hybrid potentials showing the messages computed by our algorithm.

Analogous to the situation in the previous section, we would like to perform partial maximizations of the linearized network in order to efficiently compute the maximum-likelihood estimate. Whenever a discrete variable needs to be maximized out, we encounter a partial maximization of the form

$$\mu(x) = \max_k \phi(x, d{=}k) \tag{6.14}$$

Although at first glance this may appear similar to Eq. 6.10, the discrete variable d makes this type of maximization impossible to compute efficiently. This is the case since the function $\mu(x)$ cannot efficiently be represented in closed form. Evaluating μ at a particular value of x requires keeping around the whole table of quadratic forms which define the

original hybrid potential $\phi(x, d)$. Worse, multiplying such multi-modal representations will result in a combinatorial explosion in the size of the table required for the representation. If we consider the whole constraint network, we will eventually build a table of $2^{|L|}$ Gaussians – one for every possible value of the combined set of discrete variables. This is a fundamental problems in such hybrid graphical models, with exact inference known to be NP-hard[104].

6.6.1 Approximate Hybrid Inference

While exact inference is impractical, in this section we propose using ILMS to estimate the new linearization point. This algorithm is a generalization of the algorithm proposed in Segal and Reid [171] for tracking. In that work, the approximate message passing can only deal with Markov chains, whereas we generalize to arbitrary clique trees. The idea is to approximate the maximization in Eq. 6.14 by picking the optimal value for the discrete variable d while assuming all other discrete variables in the network are fixed. Although we optimize one discrete variable at time, this maximization is performed jointly with *all* continuous variables in the rest of the network. This is possible by taking advantage of a special message passing order, different from the standard upward and downward pass.

Because finding the optimal configuration may require simultaneously toggling multiple discrete variables, our approach is only guaranteed to find a local maximum. The local nature of the approximate inference is not a huge disadvantage because we are already working with a linear approximation in the inner loop of a nonlinear optimization procedure. If the linearization is accurate, our approximation will still result in an improvement of the original objective. If the linearization is inaccurate, we are no worse off than before.

Using the clique tree of Fig. 6.3b, the approximate messages in our algorithm are com-

puted as follows:

$$\begin{aligned}
\mu_1(x) &= \max_w \Big(\phi_1(x,w)\cdot\mu_4(x)\Big) \\
d^* &= \operatorname*{argmax}_d \max_{x,y} \Big(\phi_2(x,y,d)\cdot\mu_1(x)\Big) \\
\mu_2(x) &= \max_y \phi_2(x,y,d^*) \\
\mu_3(x) &= \max_w \Big(\phi_1(x,w)\cdot\mu_2(x)\Big) \\
\mu_4(x) &= \max_z \phi_3(x,z)
\end{aligned} \tag{6.15}$$

Since computing μ_2 requires maximizing over the discrete variable d, ILMS picks the best possible value prior to computing the message. The message $\mu_1(x)$ fully captures the dependency between $\phi_2(x,y,d)$ and the rest of the network, so when we pick d^*, we are effectively picking the single Gaussian potential from $\phi_2(x,y,d)$ which is most compatible with the rest of the network. This potential then is used to compute the outgoing message μ_2. The message passing is iterated in this fashion until the discrete values stop changing. At this point we will be guaranteed to have converged to a local maximum of the CLGN in the sense that changing any single discrete variable cannot result in an improvement.

In the general case, the ILMS message passing starts at the root of the clique tree and performs a depth-first traversal of the tree sending messages along each edge as shown in Fig. 6.4. With this message order, each local update to a discrete variable will always increase the overall likelihood. In effect ILMS is performing coordinate-ascent on the space of discrete variables, but properly considering the interactions between the discrete variable being update and all continuous variables in the network.

Before each message is computed, a new value is picked for any discrete variables associated with the source clique tree node. Consider a node n with a local hybrid potential $\phi_n(x_n,d_n)$, a set of neighboring nodes $\mathtt{Ne}(n)$, and a message from node n to node s denoted by $\mu_{n\to s}$. To compute this message, we first estimate the discrete value d_n:

$$d_n^* = \operatorname*{argmax}_k \max_{x_n} \left\{ \phi_n(x_n, d_n{=}k) \left(\prod_{n'\in\mathtt{Ne}(n)} \mu_{n'\to n}(x_n) \right) \right\} \tag{6.16}$$

This means $\phi_n(x_n,d_n^*)$ is the single Gaussian potential most compatible with the rest of the network; the incoming messages capture all information about the continuous variables in

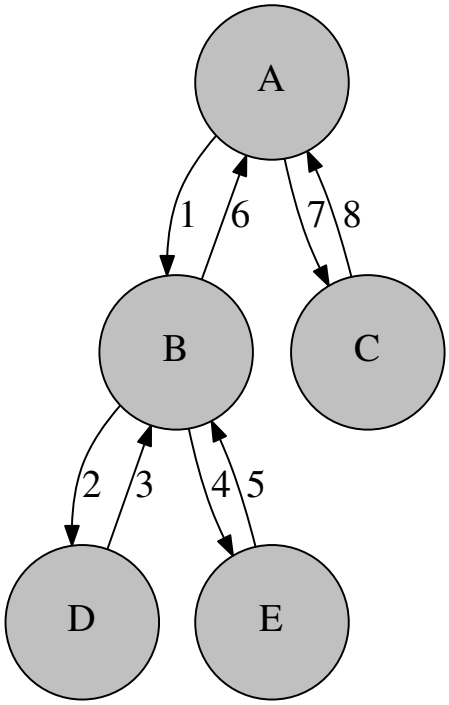


Figure 6.4: An abstract clique tree illustrating the modified message passing order required for convergence of our algorithm. Labeled nodes correspond to cliques and numbered arrows illustrate the order messages are computed in. The ordering corresponds to a depth first traversal of the tree.

the network. With the discrete variable fixed to the above value, the outgoing message is computed as

$$\mu_{n\to s} = \max_{x_n \setminus x_s} \phi_n(x_n, d_n^*) \prod_{\substack{n' \in \mathtt{Ne}(n) \\ n' \neq s}} \mu_{n'\to n}(x_n) \tag{6.17}$$

Whenever a discrete variable associated with a node changes, all messages facing away from that node become invalid since they were computed based on the previous value of the variable. However, all messages facing towards the node remain valid since there is no dependency. In the example of Fig. 6.4, consider computing message number 6. Before doing so, we must update the discrete value associated with node B, and hence invalidate all messages facing away from B in the tree, except message 6 itself which will be recomputed. With our message passing order, the next discrete variable to be updated is A (while computing message 7). This update only depends on the incoming messages to A, which are all still valid. For this reason, the message passing order suggested above guarantees the overall likelihood will always increase as each message is computed, and our approxi-

mate inference procedure always converges to a consistent estimate of all variables in the network. Unlike the standard clique tree algorithm, a single run as shown in Fig. 6.4 no longer leads to convergence; we have to keep passing messages in this cycle until a full pass through the tree occurs without any changes to the discrete variables. At this point we compute the maximum likelihood values of all the continuous variables and use this as the next linearization point.

In the evaluations below, we incorporated two additional heuristics to improve convergence. First, the message passing order is randomized where possible; when a node has multiple children, the order of their traversal is picked at random. As a second heuristic, we set the root of the clique tree to be the node with the most information so as to propagate strong observations as soon as possible. The minimum log-determinant over all information matrices in each node's table of Gaussian of potentials is used as a proxy for the information content. We pick the clique tree root once at the start of the algorithm, as soon as the initial potentials become defined after the first linearization.

6.7 Evaluation

To evaluate the proposed method, we have performed experiments comparing against three recently published algorithms for robust pose graph estimation: *Realizing, Reversing, Recovering* (RRR)[164], *Switchable Constraints* (SC)[162], and *Max-Mixtures* (MM)[160]. We have also compared against an odometry-only baseline in which all loop closure detections were ignored.

Evaluations were performed on two robust loop closure datasets provided by Latif et al. [164]. The first of these was provided together with the source code of the RRR algorithm and contains three sequences: `B25b`, `bovisa04`, and `bovisa06`. The second dataset from the same group is available online[1]; it contains 7 sequences extracted from the *KITTI*[172] dataset. The sequences are labeled as `kitti_0{N}` with non-consecutive N (see Fig. 6.5 for

[1] `htt://www.github.com/ylatif/dataset-RobustSLAM`

sequence names). All sequences provide ground truth GPS tracks as well as the output of a *Bag-of-Words* (BoW)[173] loop closure detection algorithm.

In the case of `B25b` and the *KITTI* sequences, different sets of loop closures are provided, generated by varying the threshold of the BoW algorithm. Having these variations allows the performance of the robust loop closure algorithms to be tested in conditions ranging from a few very confident loop closures to many loop closures of questionable accuracy. `B25b` contain loops closure data with 41 different values of the threshold; the *KITTI* sequences provide 21 different values. Between the different sequences and sets of loop closures, this makes for 190 variations on which the algorithms have been tested.

To run the evaluations, we used the publicly available version of the RRR algorithm available on the authors' website, with the suggested parameters for each dataset. For the *Switchable Constraints* and *Max-Mixtures* algorithms, we used publicly available implementations provided by Sünderhauf and Protzel [81] with default parameters. For our own *Hybrid Inference Optimization* (HIO) algorithm, we used input objectives and constants identical to the *Max-Mixtures* defaults. Specifically, we used a prior of 0.01 for outliers and set up the outlier objective information matrix to be 10^{-12} times the information matrix of the corresponding inlier objective. The same parameters were used for all evaluations presented.

The overall quality of the reconstructed trajectory was evaluated using the Absolute Trajectory Error (ATE) as defined in the Rawseeds[174] toolkit. The ATE measures the average XY distance between the reconstructed trajectory and the ground truth data after the two have been put into optimal alignment.

Our main results for ATE are shown in Fig. 6.5. For each sequence, we plot the median ATE error achieved over the range of BoW thresholds; the error bars show the minimum and maximum values. The median, min, and max are more meaningful statistics than the mean and standard deviation since gross failures of the algorithms often result in exceptionally large values of the ATE which skew the mean. Fig. 6.6 shows a more detailed

view of the individual ATE scores for `B25b` as a function of the BoW threshold. The results show that the *Hybrid Inference Optimization* algorithm outperforms the others on almost all of the sequences. Notably, our overall worst-case performance is dramatically better. The RRR algorithm is interesting in this sense because it often performs as well as HIO, but when it fails it does so catastrophically.

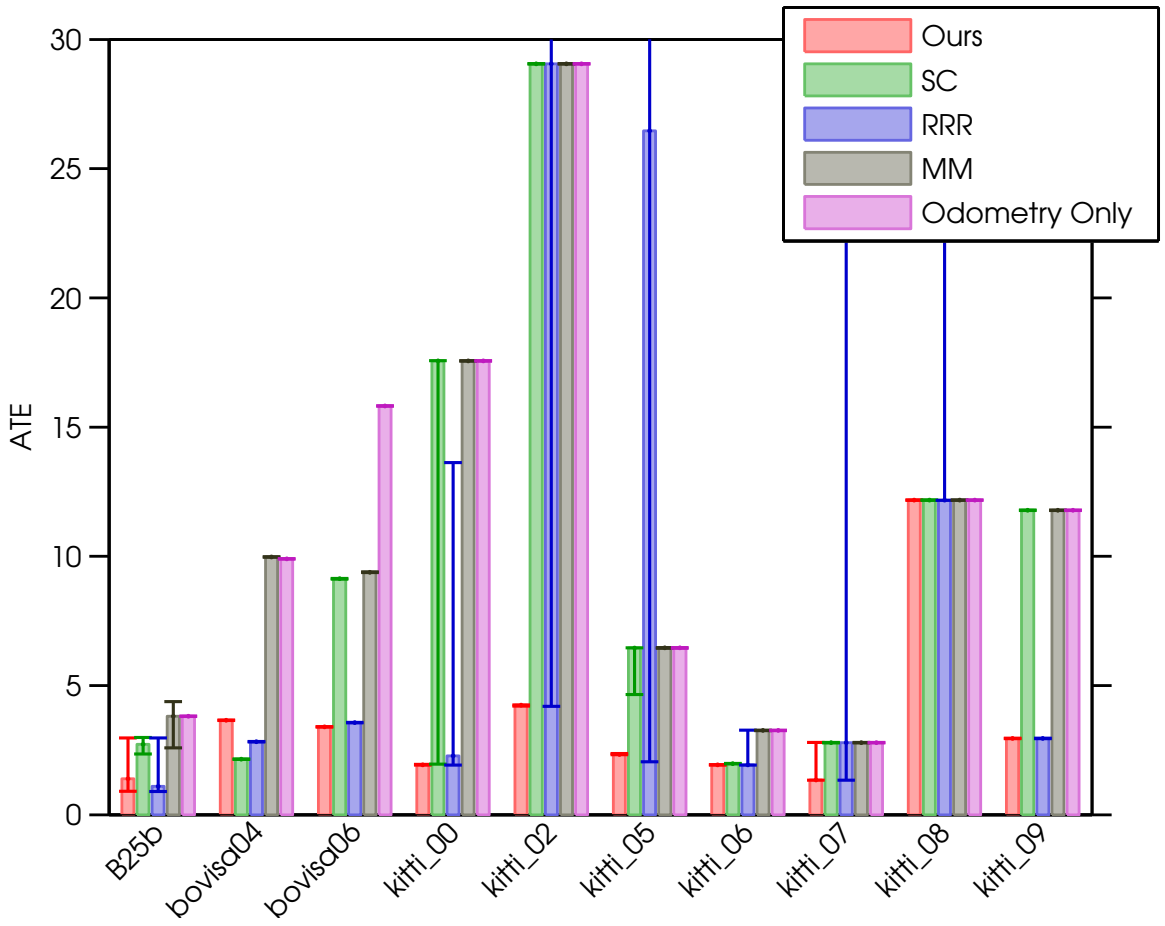


Figure 6.5: Evaluation results on 10 different sequences. We show the median ATE along with the minimum and maximum values as error bars. Note that the `bovisa04` and `bovisa06` datasets only provide one sequence, so the min, max, and median all coincide. The `Odometry Only` algorithm is the baseline ATE obtained by running `g2o` with all loop closures removed.

For the *KITTI* sequences, where ground truth loop closures are available, we have also looked at the number of incorrect loop closures accepted by the algorithms and the number of correct loop closures which were erroneously discarded. Although this information is equivalent to the commonly used Precision/Recall metrics, it is more informative to look at

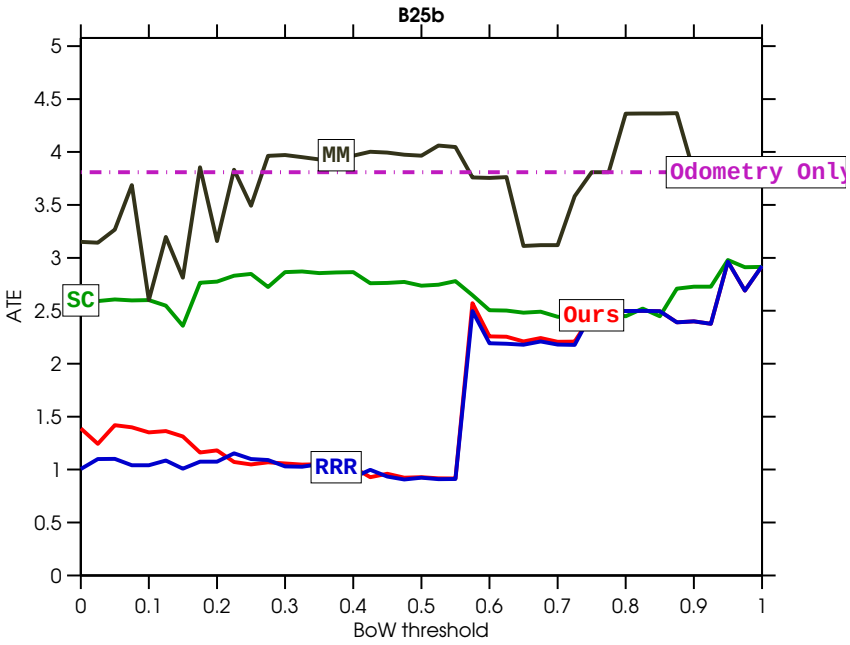


Figure 6.6: ATE as a function of the BoW threshold on the `B25b` sequence.

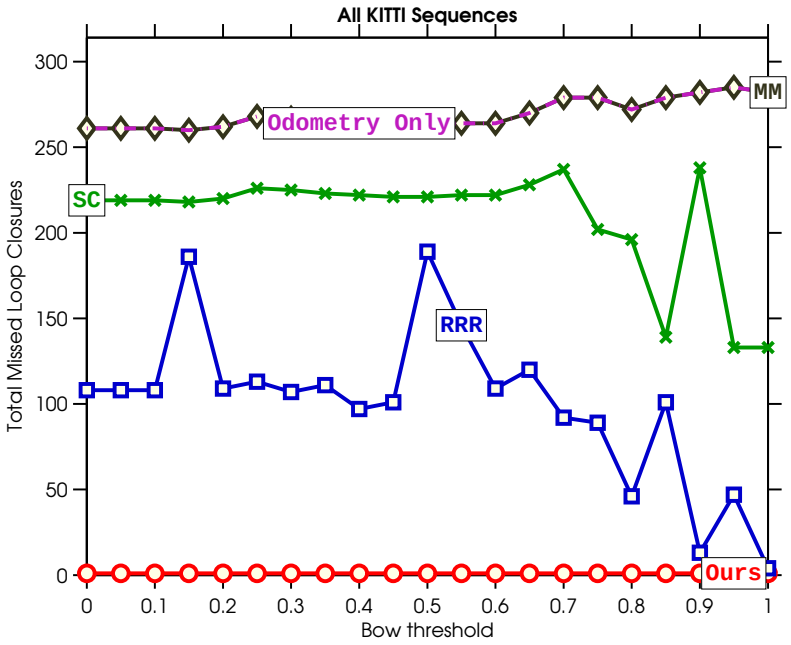
the number of individual loops. Complete failure of the algorithms can often be caused by a single incorrect loop closure, so it is useful to look at the results with higher granularity than provided by the Precision/Recall ratios. Fig. 6.7 shows a plot of these counts as a function of the BoW threshold aggregated across all of the *KITTI* sequences. For the *KITTI* sequences, our algorithm correctly finds almost all of the annotated ground truth loop closures with no false positives and only one false negative in the `kitti_05` sequence. In this sense, our performance on these sequences (Fig. 6.5) is close optimal.

It is noteworthy that the *Max-Mixtures* algorithm does not perform significantly better than the baseline in our evaluations. This corresponds with the poor performance of the algorithm measured by Sünderhauf and Protzel [81], but not to the original results published by Olson and Agarwal [160]. We believe this disparity is caused by different evaluation protocols. Olson and Agarwal evaluated performance using an incremental evaluation scheme with optimization being run online. In our evaluation, the system state was initialized to the trajectory indicated by the odometry with no loop closures. It is possible that this initialization corresponds to a local minimum for *Max-Mixtures*.

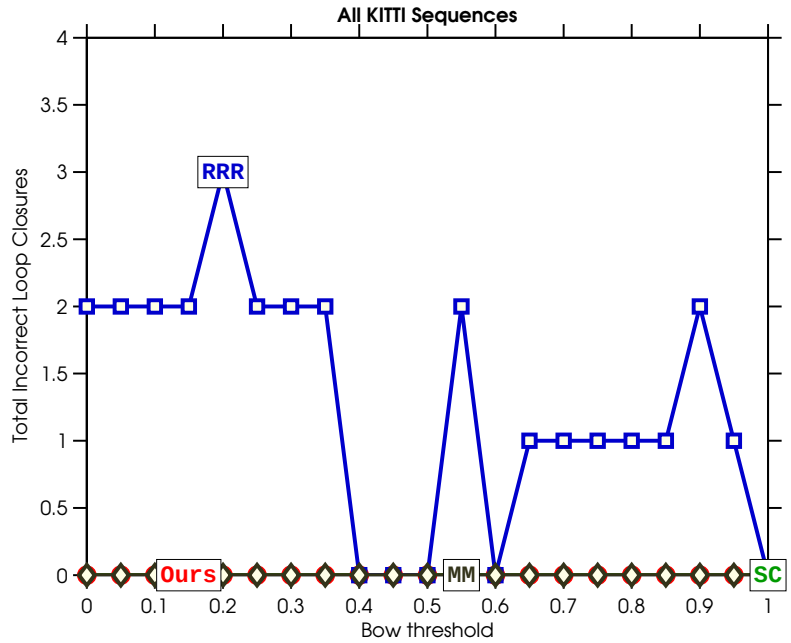
The improved performance of *Hybrid Inference Optimization* comes at a cost in terms of speed. Tab. 6.1 shows statistics over the running times of the algorithms in our evaluation. HIO is the slowest, with RRR coming in second-slowest. For offline and batch processing, we argue that the increased accuracy provided by HIO more than makes up for the extra processing time. In the real-time use case, however, another algorithm may be more suitable.

Algorithm	min (s)	median (s)	mean (s)	max (s)
Ours	0.22	2.70	4.38	19.89
SC	0.09	0.51	0.48	1.21
RRR	0.04	0.77	0.99	6.03
MM	0.09	0.47	0.45	1.19

Table 6.1: The mean, median, min, and max running times over all tests.



(a) Missed Loop Closures. The `Odometry Only` measurement provides a baseline for worst-case performance, e.g. all loop closures missed.



(b) Incorrect Loop Closures. The baseline `Odometry Only` measurement is not shown since it is not relevant.

Figure 6.7: Comparison of missed loop closures and incorrect loop closures detected by the various algorithms.

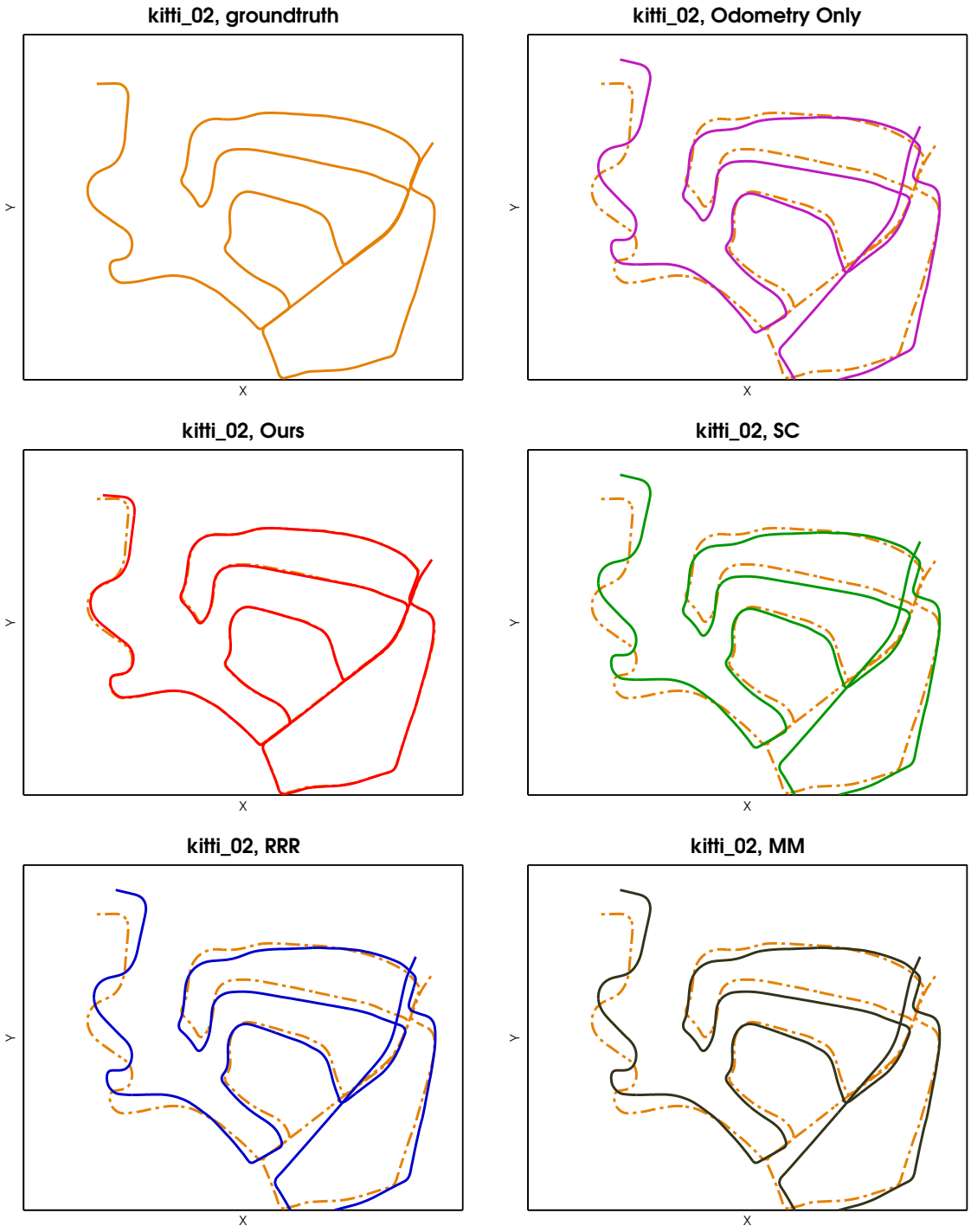


Figure 6.8: The trajectory output of the algorithms we evaluated alongside groundtruth (dashed/orange) and raw odometry tracks for the kitti_02 test sequence. Our approach correctly identifies the loop closures.

6.8 Conclusions

Taking into account discrete variables in large scale SLAM problems is likely to become increasingly important as these systems are expected to operate over larger environments unassisted. This requires the development of large scale discrete/continuous optimization algorithms which can handle multimodality in the objective. We have proposed such an algorithm and have evaluated it on the problem of robust pose graph estimation. Our results show a large improvement over competing methods on this particular problem, and the general nature of the technique suggests that it may have broader applicability in the robotics community.

Acknowledgment

We gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council [grant number EP/H050795], the Australian Research Council [grant DP130104413, Laureate Fellowship FL130100102 to IDR, and the Centre of Excellence in Robotic Vision, CE140100016], as well as helpful conversations with Cesar Cadena.

7

Conclusions and Future Work

7.1 Conclusions

The research described in this thesis has been motivated by the long-term goal of enabling intelligent perception in robotics. Despite much progress in large scale mapping, the current state of the art still relies heavily on the static world assumption and various preprocessing heuristics. Long term and robust autonomy requires an estimation framework capable of dealing with changing environments, ambiguous sensor data, and interpretation of the environment beyond mere geometry. To this end, we have presented a framework capable of combining discrete decision making with large scale continuous state estimation.

Our framework models these discrete/continuous problems as Conditional Linear Gaussian Networks and uses a novel approximate inference technique which we call Iterative Local Model Selection (ILMS). In Chapter 4, we formally described the ILMS algorithm and validated its performance against competing deterministic inference techniques. Following this, we presented two novel algorithms based on the ILMS strategy. Chapter 5 applied ILMS to multi-target tracking of pedestrians by recasting data association as a Switching Linear Dynamical System. Chapter 6 used ILMS for Robust Pose Graph Optimization by explicitly modeling outlier loop closures with binary indicator variables. Despite targeting different areas of perception, both algorithms were able to either match or exceed the state

of the art. This fact is both promising and somewhat surprising given the greedy nature of the algorithm.

We believe the key to understanding why the ILMS strategy works is looking at the algorithm through the lens of a block coordinate ascent procedure. Each discrete variable update is actually performing an optimization over the state of the discrete variable itself, but also the state of *all* continuous variables in the system. This turns out to be a surprisingly effective optimization strategy, partly because it matches the way the variables tend to be naturally correlated. Discrete model selection variables tend to be the most correlated with the continuous variables that they 'switch'. This makes it important to always keep them coupled during any optimization – something which the Variational Bayes [103] algorithm explicitly avoids.

7.2 Future Work

The simplicity of ILMS and its performance on the applications addressed in this thesis suggests some interesting topics for future research. There are several low hanging fruit based on combining combining various elements of the research presented in this thesis. For example, the Latent Data Association algorithm in Chapter 5 can be applied non-linear settings using the re-linearization scheme from Chapter 6. The tracker could also be extended to include mixtures of different motion models for the objects being tracked, similar to the experiments in Chapter 4.

Another possible extension is a joint tracking and mapping system. Unfortunately, uncertainty in the camera position induces correlations between the states of the tracked objects, but the data association strategy in Chapter 5 requires these states to be uncorrelated. If this is not the case, the score for a given data association cannot be decomposed as the sum of individual track scores, making it impossible to use a Linear Assignment Problem. One potential solution would be to use the JCBB [78] algorithm discussed in Section 2.5.1 instead. JCBB Data Association is specifically designed to address the prob-

lem of correlated landmarks, but is slower than simpler methods.

The local nature of the algorithm also allows for some more interesting models to be shoe-horned into the CLGN framework. Since ILMS does not require the space of discrete variables to be easily enumerable, there is nothing preventing us from choosing among an extremely large, or even infinite, set of local potentials. For example, in Chapter 5, the Linear Assignment Problem was used to select from the set of all possible data associations at each frame. This concept can be taken further by allowing approximate and non-deterministic selection schemes which pick a model from a large structured space.

One can imagine directly selecting image features as part of the estimation process in visual SLAM. In this case we would pick image features by associating image points with the hypothetical resulting observation potential. Feature selection could be performed by picking the best such potential, and would automatically take into account the current prior on landmark locations. More ambitiously, the same idea can be applied to pedestrian detection and tracking. Starting with the Latent Data Association algorithm in Chapter 5, we can imagine looking for pedestrian detections at time t which are compatible with the predicted locations of tracked targets, as communicated by the incoming messages from $t-1$ and $t+1$. Such a combined tracking and detection algorithm is very appealing because it could be used to bootstrap object detectors and classifiers directly from video sequence without requiring manual annotation of every single frame. The Latent SVM machinery used by Felzenszwalb et al. [175] makes this detector particularly appealing because the classification procedure already involves maximization over a discrete set.

Another possible topic for future work is a characterization of the accuracy of the ILMS local posterior uncertainty. While accurate decision making is important for robotics applications, it is also critical to use algorithms which are able to gauge their own confidence in order to avoid catastrophic mistakes[176]. The ILMS algorithm naturally provides a local estimate of the posterior distribution $\mathbf{P}(x_i, d_i \mid d_{\text{other}})$. Each such local posterior, however, is conditional on a fixed value of all other discrete variables aside from d_i. This suggests

the possibility of overconfidence since the posteriors do not properly take into account alternative *discrete* explanations for the data. Whether or not this approximation is accurate depends on the specifics of the application. For situations where the majority of the probability mass is concentrated in a single discrete mode, the ILMS posteriors can be accurate since uncertainty in the continuous variables is properly taken into account. For more ambiguous situations, the approximations involved are likely to make the algorithm overconfident. That said, an MCMC version of the algorithm could be constructed via Gibbs sampling which would mitigate these problems. In fact, the ILMS algorithm is particularly well suited for this — replacing the maximization step with sampling over the marginal distribution

$$\mathbf{P}(d_i \mid d_{\text{other}}) = \int \mathbf{P}(d_i, x_i \mid d_{\text{other}}) \, \mathrm{d}x_i \tag{7.1}$$

would result in a Collapsed Gibbs sampler.

A

Appendix: Iterative Local Model Selection

A.1 Proof of Convergence

Proposition 2. The general version of the Iterative Local Model Selection algorithm in Fig. 4.6 will always converge.

Proof. As in Prop. 1, assume that at least one forward and backward pass of the algorithm has been completed so that all messages and local beliefs have been updated at least once. Consider the table of values v_{ij} computed in the forward pass as shown in Eq. 4.48 and on line 38 of Fig. 4.6. Let $v_t(i,j) = v_{ij}$ refer to this table at time t in the forward pass. At this point ILMS will perform the following steps

1. Update the local belief $\mathcal{B}_t$
2. Update the table v_t
3. Update $d^*_{t-1}\big|_{d_t}$
4. Update $\overrightarrow{\mu}_t$
5. Update the local belief $\mathcal{B}_{t+1}$
6. Update the table v_{t+1}

7. Update $d_t^*|_{d_{t+1}}$

8. Update $\overrightarrow{\mu}_{t+1}$

We want to show that the value of $v_t^* = \max_{ij}\{v_{ij}\}$ can only increase as the forward message passing progresses. To see this, consider the situation after step 6. At this point we use the definition of v_t and expand the incoming message $\overleftarrow{\mu}_{t+1}$

$$\begin{aligned} v_t(i,j) &= \mathbf{marg}\left[\mathcal{B}_t^{(i,j)}\right] = \\ &= \mathbf{marg}\left[\overrightarrow{\mu}_{t-1}^{(i)} \cdot \phi_t^{(i,j)} \cdot \overleftarrow{\mu}_{t+1}^{(j)}\right] \\ &= \mathbf{marg}\left[\overrightarrow{\mu}_{t-1}^{(i)} \cdot \phi_t^{(i,j)} \cdot \phi_{t+1}^{(j,d_{t+1}^*|_{d_t=j})} \cdot \overleftarrow{\mu}_{t+2}^{(d_{t+1}^*|_{d_t=j})}\right] \end{aligned} \tag{A.1}$$

Since $d_{t-1}^*\big|_{d_t}$ has just been updated using Eq. 4.49, we also know that

$$\max_{ij} v_t(i,j) = \max_j \left\{ v_t(\, d_{t-1}^*|_{d_t=j}, j)\right\} \tag{A.2}$$

Expanding the right hand side of the above using Eq. A.1 we get that

$$\max_{ij}\{v_t(i,j)\} = \max_j \left\{ \mathbf{marg}\left[\overrightarrow{\mu}_{t-1}^{(d_{t-1}^*|_{d_t=j})} \cdot \phi_t^{(d_{t-1}^*|_{d_t=j}, j)} \cdot \phi_{t+1}^{(j,d_{t+1}^*|_{d_t=j})} \cdot \overleftarrow{\mu}_{t+2}^{(d_{t+1}^*|_{d_t=j})}\right]\right\} \tag{A.3}$$

We can also write down the definition of v_{t+1} with the incoming message $\overrightarrow{\mu}_t$ expanded

$$\begin{aligned} v_{t+1}(j,k) &= \mathbf{marg}\left[\mathcal{B}_{t+1}^{(j,k)}\right] = \\ &= \mathbf{marg}\left[\overrightarrow{\mu}_t^{(j)} \cdot \phi_{t+1}^{(j,k)} \cdot \overleftarrow{\mu}_{t+2}^{(k)}\right] \\ &= \mathbf{marg}\left[\overrightarrow{\mu}_{t-1}^{(d_{t-1}^*|_{d_t=j})} \cdot \phi_t^{(d_{t-1}^*|_{d_t=j}, j)} \cdot \phi_{t+1}^{(j,k)} \cdot \overleftarrow{\mu}_{t+2}^{(k)}\right] \end{aligned} \tag{A.4}$$

In this case, observe that $d_{t+1}^*\big|_{d_t}$ was updated using Eq. 4.51 during the backward pass so it is not up to date with the newly calculated values in v_{t+1}. This means that

$$\max_{jk}\{v_{t+1}(j,k)\} \geq \max_j \left\{ v_{t+1}(j,\, d_{t+1}^*|_{d_t=j})\right\} \tag{A.5}$$

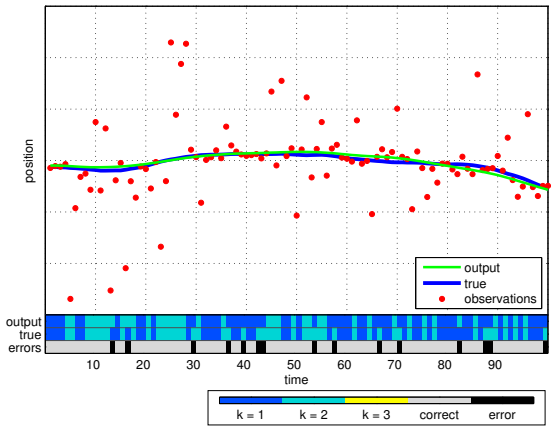
This time we expand v_{t+1} on the right hand side using Eq. A.4 to get

$$\max_{jk}\{v_{t+1}(j,k)\} \geq \max_j \left\{ \mathbf{marg}\left[\overrightarrow{\mu}_{t-1}^{(d_{t-1}^*|_{d_t=j})} \cdot \phi_t^{(d_{t-1}^*|_{d_t=j}, j)} \cdot \phi_{t+1}^{(j,d_{t+1}^*|_{d_t=j})} \cdot \overleftarrow{\mu}_{t+2}^{(d_{t+1}^*|_{d_t=j})}\right]\right\} \tag{A.6}$$

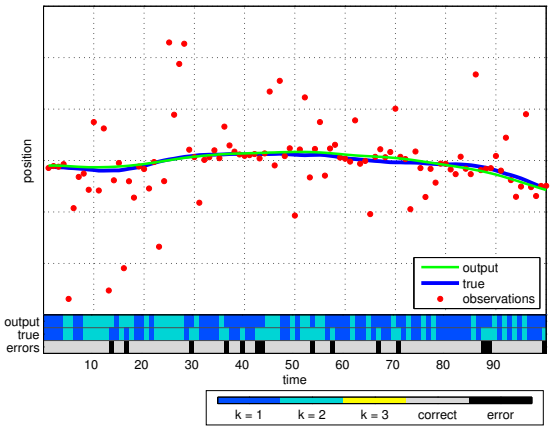
Inspecting the right hand sides of Eq. A.6 and Eq. A.3 we can see that they are identical. This shows that $v_{t+1}^* = \max_{jk}\{v_{t+1}(j,k)\} \geq \max_{ij}\{v_t(i,j)\} = v_t^*$ and so v_t^* must increase

as the forward message passing progresses. The backward message passing can be shown to have the same property in the opposite direction ($v_T^* \leq v_{T-1}^* \leq \cdots \leq v_1^*$), and so iterative applications of forward and backward message passing must converge at some point when the values of v_t^* cannot be increased any further. □

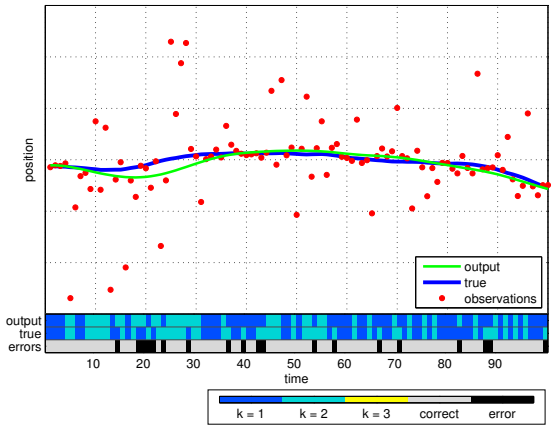
A.2 Additional Figures and Results



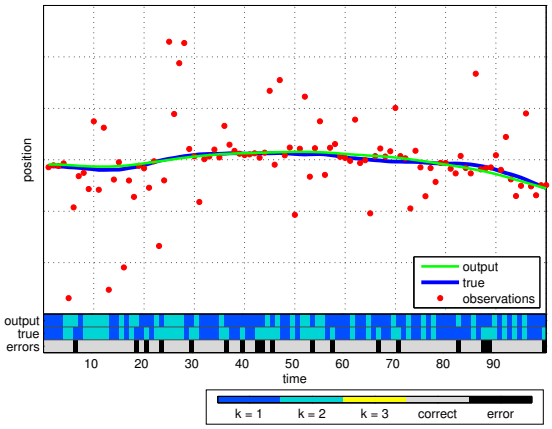
(a) ILMS_MM



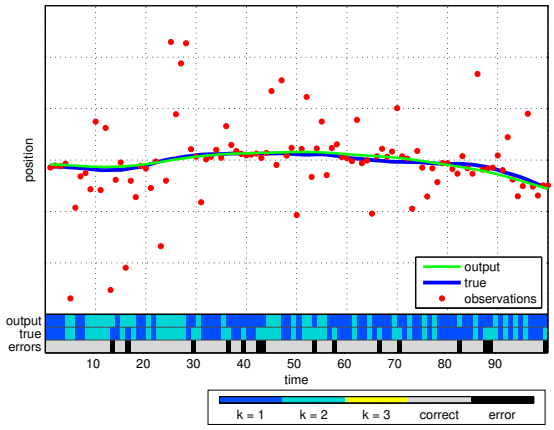
(b) ILMS_ML



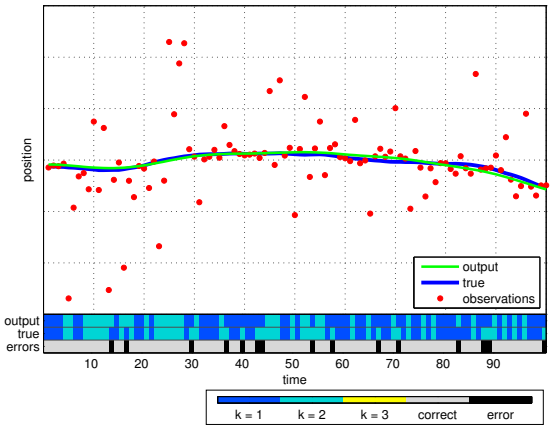
(c) AV



(d) GPB2S

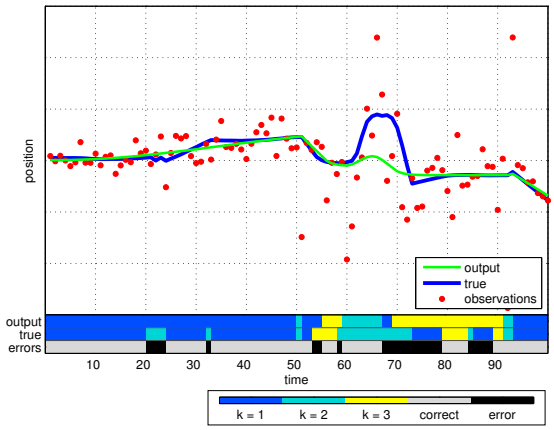


(e) VB

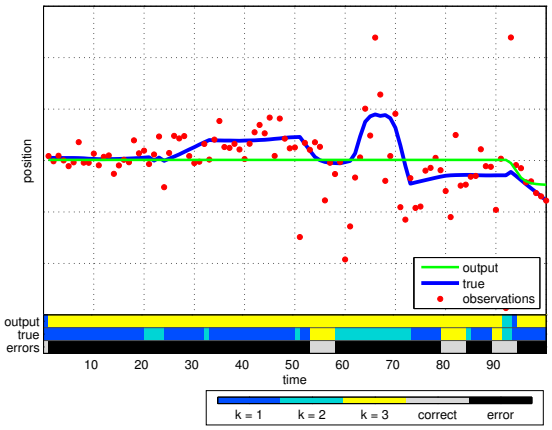


(f) EP

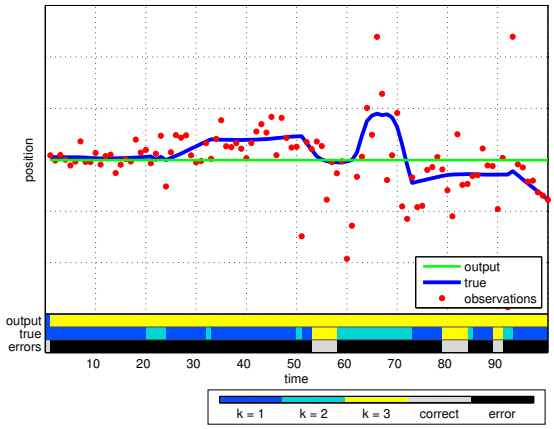
Figure A.1: Sampled trajectories and results from experiment `OUT` with random outlier measurements. Each plot shows the true trajectory in blue, the observations in red, and the estimated trajectory in green. Below the trajectory data, the true and estimated discrete states are shown as colored bars. $k = 1$ corresponds to inlier observations and $k = 2$ to outliers. $k = 3$ is not applicable in this experiment. A third bar indicates discrete estimation errors in black.



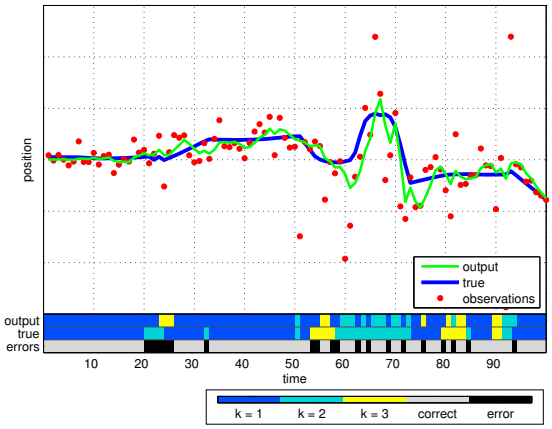
(a) ILMS_MM



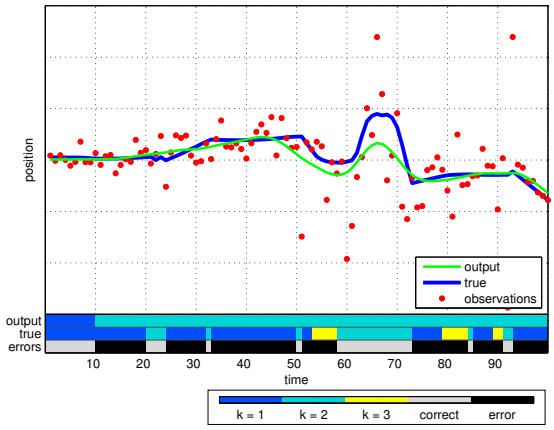
(b) ILMS_ML



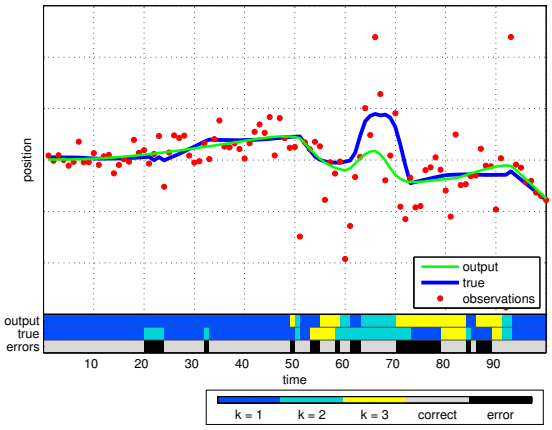
(c) AV



(d) GPB2

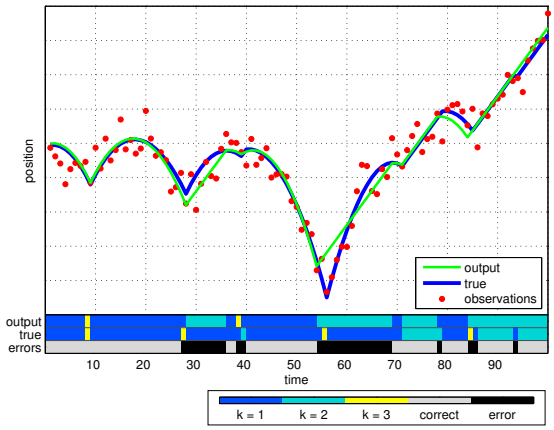


(e) VB

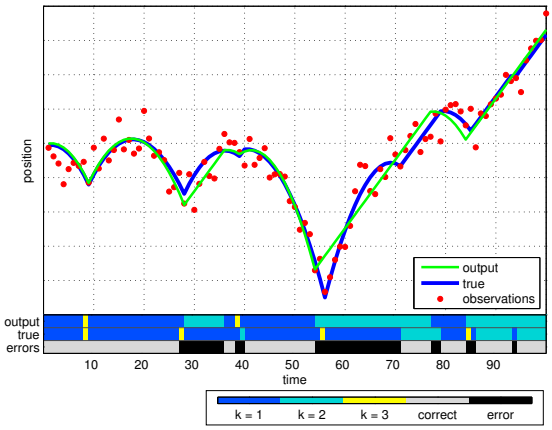


(f) EP

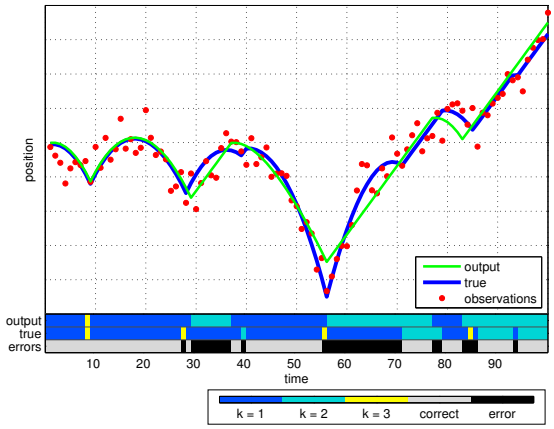
Figure A.2: Sampled trajectories and results from experiment `MNV1` with correlated outlier measurements. Each plot shows the true trajectory in blue, the observations in red, and the estimated trajectory in green. Below the trajectory data, the true and estimated discrete states are shown as colored bars. $k = 1$ corresponds to inlier observations and $k = 2$ to outliers. $k = 3$ is not applicable in this experiment. A third bar indicates discrete estimation errors in black.



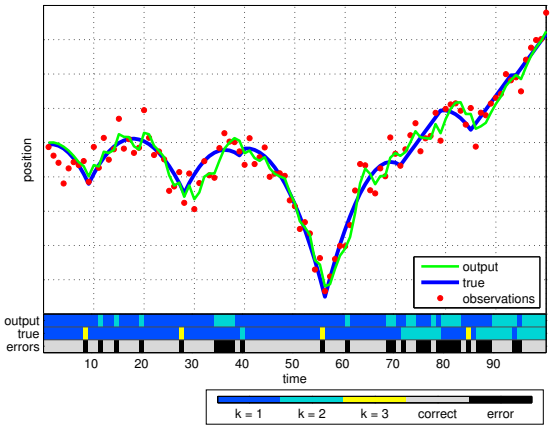
(a) ILMS_MM



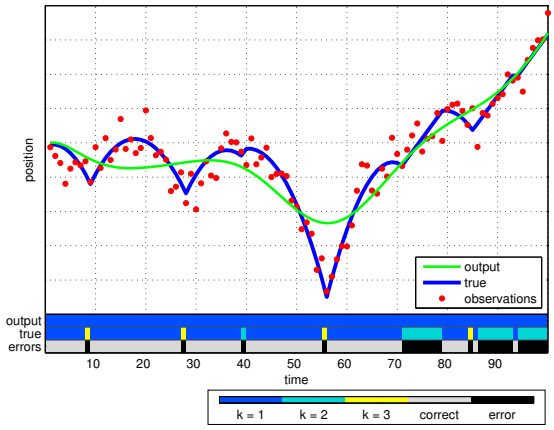
(b) ILMS_ML



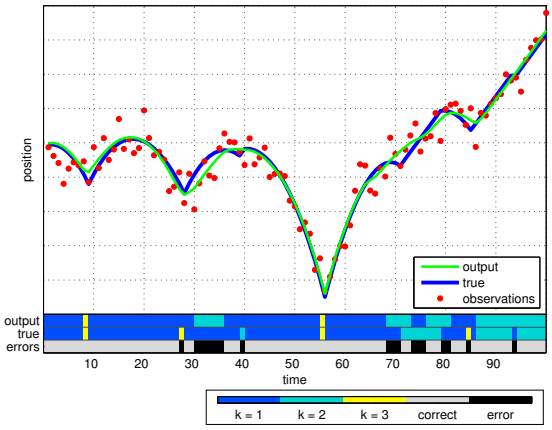
(c) AV



(d) GPB2

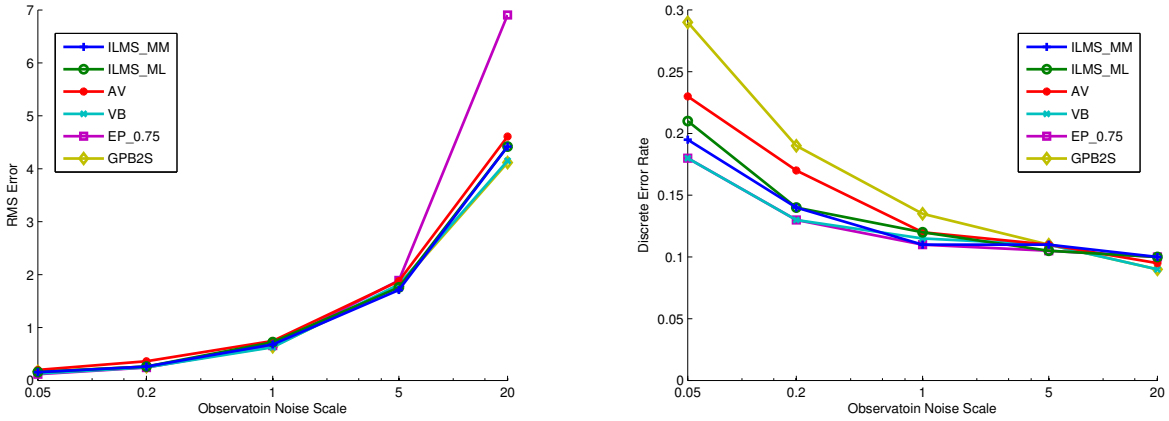


(e) VB

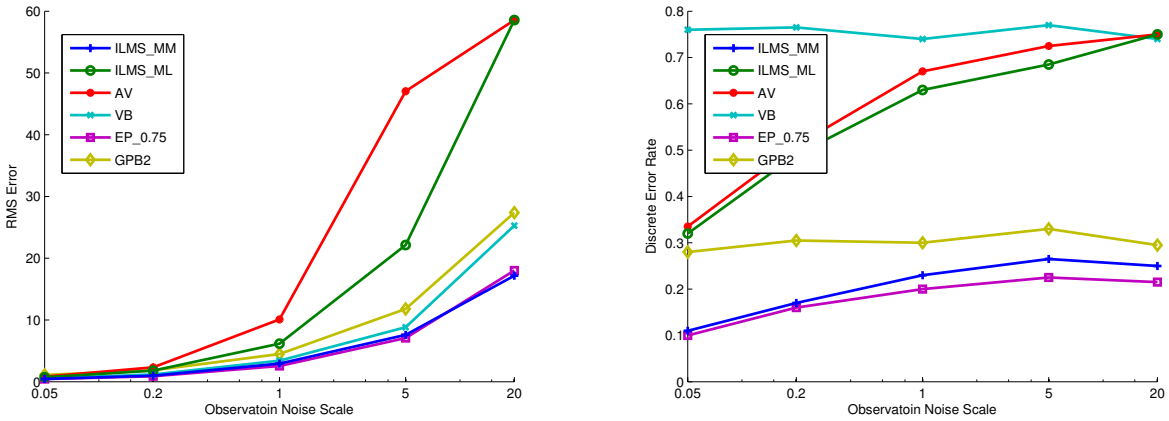


(f) EP

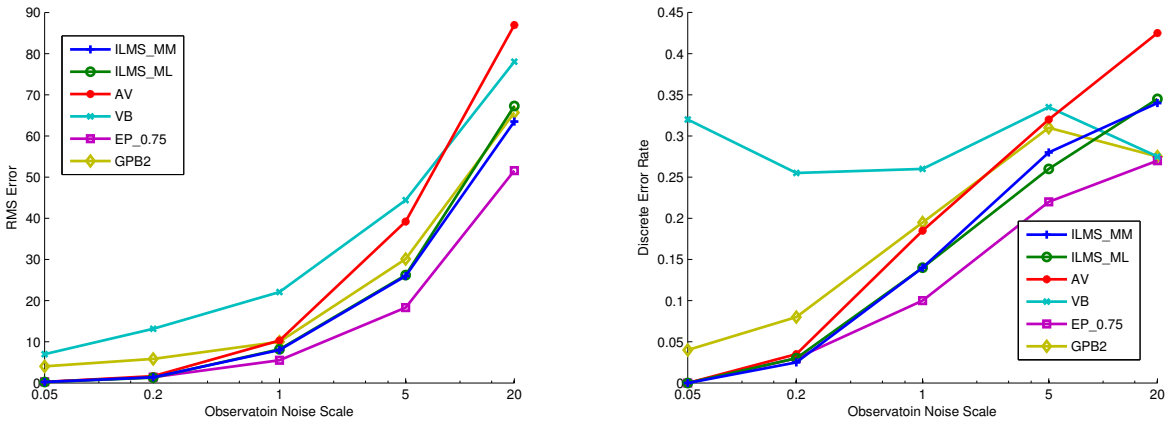
Figure A.3: Sample output from the `MNV2` experiment with three different regimes corresponding to smooth motion ($k = 1$), maneuvering ($k = 2$), and slowing down ($k = 3$). Each plot shows the true trajectory in blue, the observations in red, and the estimated trajectory in green. Below the trajectory data, the true and estimated discrete states are shown as colored bars. A third bar indicates discrete estimation errors in black.



(a) OUT



(b) MNV1



(c) MNV2

Figure A.4: Algorithm performance with varying levels of observation noise for the three experiments. Median RMS Error in the estimated position is shown on the left and median discrete error rate on the right.

A.3 Dataset Parameters

Tab. A.1 lists the synthetic dataset parameters corresponding to the SDLS model in Chapter 4. This model is repeated below for convenience:

$$d_t \in \{1, \ldots, K\}, x_t \in \mathbb{R}^N, z_t \in \mathbb{R} \tag{A.7}$$

$$d_1 \sim \mathcal{D}(p_0) \tag{A.8}$$

$$d_t \mid d_{t-1}{=}i \sim \mathcal{D}(\tau_i) \qquad t > 1 \tag{A.9}$$

$$x_1 \sim \mathcal{N}(\mathbf{0}, \Sigma_0) \tag{A.10}$$

$$x_t \mid x_{t-1}, d_t{=}j \sim \mathcal{N}\left(A^{(j)} x_{t-1} + b^{(j)}, \Sigma_{\text{mot}}^{(j)}\right) \qquad t > 1 \tag{A.11}$$

$$z_t \mid x_t, d_t{=}j \sim \mathcal{N}\left(G x_t, \Sigma_{\text{obs}}^{(j)}\right) \tag{A.12}$$

Parameter	`OUT`	`MNV1`	`MNV2`
K	2	3	3
N	2	2	3
p_0	$\begin{bmatrix}0.5 & 0.5\end{bmatrix}^\top$	$\begin{bmatrix}0.98 & 0.01 & 0.01\end{bmatrix}^\top$	$\begin{bmatrix}0.998 & 0.001 & 0.001\end{bmatrix}^\top$
τ_1	$\begin{bmatrix}0.5 & 0.5\end{bmatrix}^\top$	$\begin{bmatrix}0.85 & 0.05 & 0.10\end{bmatrix}^\top$	$\begin{bmatrix}0.950 & 0.050 & 0.050\end{bmatrix}^\top$
τ_2	$\begin{bmatrix}0.5 & 0.5\end{bmatrix}^\top$	$\begin{bmatrix}0.30 & 0.70 & 0.00\end{bmatrix}^\top$	$\begin{bmatrix}0.100 & 0.900 & 0.000\end{bmatrix}^\top$
τ_3	–	$\begin{bmatrix}0.00 & 0.20 & 0.80\end{bmatrix}^\top$	$\begin{bmatrix}1.000 & 0.000 & 0.000\end{bmatrix}^\top$
Σ_0	$\mathbf{diag}\begin{bmatrix}1 & 10^{-6}\end{bmatrix}$	$\mathbf{diag}\begin{bmatrix}1 & 0.05^2\end{bmatrix}$	$\mathbf{diag}\begin{bmatrix}1 & 10^{-6} & 10^{-6}\end{bmatrix}$
$A^{(1)}$	$\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & 1 & \frac{1}{2}\\ 0 & 1 & 1\\ 0 & 0 & 1\end{bmatrix}$
$b^{(1)}$	$\mathbf{0}$	$\mathbf{0}$	$\begin{bmatrix}0 & -2 & 0\end{bmatrix}^\top$
$\Sigma^{(1)}_{\text{mot}}$	$\mathbf{diag}\begin{bmatrix}10^{-6} & 10^{-2}\end{bmatrix}$	$\mathbf{diag}\begin{bmatrix}1 & 10^{-6}\end{bmatrix}$	$\mathbf{diag}\begin{bmatrix}10^{-6} & 10^{-6} & 10^{-2}\end{bmatrix}$
$\Sigma^{(1)}_{\text{obs}}$	2^2	5^2	20^2
$A^{(2)}$	$= A^{(1)}$	$= A^{(1)}$	$\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$
$b^{(2)}$	$\mathbf{0}$	$\mathbf{0}$	$\begin{bmatrix}10 & 0 & 0\end{bmatrix}^\top$
$\Sigma^{(2)}_{\text{mot}}$	$= \Sigma^{(1)}_{\text{mot}}$	$\mathbf{diag}\begin{bmatrix}10^{-6} & 2.5^2\end{bmatrix}$	$= \Sigma^{(1)}_{\text{mot}}$
$\Sigma^{(2)}_{\text{obs}}$	20^2	25^2	$= \Sigma^{(1)}_{\text{obs}}$
$A^{(3)}$	–	$\begin{bmatrix}1 & 1\\ 0 & 0.5\end{bmatrix}$	$\begin{bmatrix}1 & 1 & \frac{1}{2}\\ 0 & -1 & -1\\ 0 & 0 & 0\end{bmatrix}$
$b^{(3)}$	–	$\mathbf{0}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
$\Sigma^{(3)}_{\text{mot}}$	–	$\mathbf{diag}\begin{bmatrix}10^{-6} & 10^{-6}\end{bmatrix}$	$= \Sigma^{(1)}_{\text{mot}}$
$\Sigma^{(3)}_{\text{obs}}$	–	10^2	$= \Sigma^{(1)}_{\text{obs}}$

Table A.1: Parameter values for the synthetic experiments in Chapter 4.

Bibliography

[1] S. Agarwal, N. Snavely, I. Simon, S. M Seitz, and R. Szeliski. Building rome in a day. In *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79. IEEE, October 2009.

[2] Kai Ni, D. Steedly, and F. Dellaert. Tectonic SAM: exact, Out-of-Core, Submap-Based SLAM. In *2007 IEEE International Conference on Robotics and Automation*, pages 1678–1685. IEEE, April 2007.

[3] Ananth Ranganathan, Michael Kaess, and Frank Dellaert. Loopy sam. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6–12, 2007.

[4] Charles Bibby and Ian Reid. Simultaneous localisation and mapping in dynamic environments (slamide) with reversible data association. In *Proceedings of Robotics: Science and Systems*, 2007.

[5] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASMEJournal of Basic Engineering*, 82(1):3545, 1960.

[6] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, 2003.

[7] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[8] P. Kaminski, Arthur E. Bryson, and S. Schmidt. Discrete square root filtering: A survey of current techniques. *Automatic Control, IEEE Transactions on*, 16(6):727–736, 1971.

[9] M. Morf and T. Kailath. Square-root algorithms for least-squares estimation. *Automatic Control, IEEE Transactions on*, 20(4):487–497, Aug 1975.

[10] P. Dyer and S. McReynolds. Extension of square-root filtering to include process noise. *Journal of Optimization Theory and Applications*, 3(6):444–458, 1969.

[11] Yaakov Bar-Shalom, Xiao-Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation*. John Wiley and Sons, June 2001.

[12] RAUCH H. E., STRIEBEL C. T., and TUNG F. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965. doi: 10.2514/3.3166.

[13] D. Fraser and J. Potter. The optimum linear smoother as a combination of two optimum linear filters. *Automatic Control, IEEE Transactions on*, 14(4):387–390, Aug 1969.

[14] H. Cox. On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Transactions on Automatic Control*, 9(1):5– 12, January 1964.

[15] Bruce A McElhoe. An assessment of the navigation and course corrections for a manned flyby of mars or venus. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-2(4):613–623, July 1966.

[16] B. M Bell and F. W Cathey. The iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, February 1993.

[17] B. Bell. The iterated Kalman smoother as a Gauss-Newton method. *SIAM Journal on Optimization*, 4(3):626–636, 1994.

[18] D. Bertsekas. Incremental least squares methods and the extended kalman filter. *SIAM Journal on Optimization*, 6(3):807–822, 1996.

[19] Simon Julier and Jeffrey K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, University of Oxford, 1996.

[20] Simon J. Julier and Jeffrey K. Uhlmann. New extension of the kalman filter to nonlinear systems, 1997.

[21] Gabe Sibley, Gaurav Sukhatme, and Larry Matthies. The iterated sigma point kalman filter with applications to long range stereo. In *Robotics: Science and Systems*, volume 8, pages 235–244, 2006.

[22] D. Magill. Optimal adaptive estimation of sampled stochastic processes. *Automatic Control, IEEE Transactions on*, 10(4):434–439, Oct 1965.

[23] G. Ackerson and K.S. Fu. On state estimation in switching environments. *Automatic Control, IEEE Transactions on*, 15(1):10–17, Feb 1970.

[24] Jitendra K. Tugnait. Detection and estimation for abruptly changing systems. *Automatica*, 18(5):607 – 615, 1982.

[25] H. A P Blom. An efficient filter for abruptly changing systems. In *Decision and Control, 1984. The 23rd IEEE Conference on*, pages 656–658, Dec 1984.

[26] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*. Mathematics in Science and Engineering Series. Academic Press, 1988.

[27] Yaakov Bar-Shalom and Edison Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, September 1975.

[28] Y Bar-Shalom, T Fortmann, and M Scheffe. Joint probabilistic data association for multiple targets in clutter. In *Proc. Conf. on Information Sciences and Systems*, pages 404–409, 1980.

[29] T.E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Multi-target tracking using joint probabilistic data association. In *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, pages 807–812, Dec 1980.

[30] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173– 184, July 1983.

[31] J.B. Collins and J.K. Uhlmann. Efficient gating in data association with multivariate gaussian distributed states. *Aerospace and Electronic Systems, IEEE Transactions on*, 28 (3):909–916, Jul 1992.

[32] Songhwai Oh, S. Russell, and S. Sastry. Markov chain monte carlo data association for general multiple-target tracking problems. In *43rd IEEE Conference on Decision and Control, 2004. CDC*, volume 1, pages 735–742 Vol.1. IEEE, December 2004.

[33] R. J Fitzgerald. Track biases and coalescence with probabilistic data association. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21(6):822–825, November 1985.

[34] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843– 854, December 1979.

[35] Ingemar J. Cox and Sunita L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[36] Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, 2004.

[37] R.L. Streit and T.E. Luginbuhl. Probabilistic multi-hypothesis tracking. Technical report, DTIC Document, 1995.

[38] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[39] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

[40] Philippe Moutarlier and Raja Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th Int. Symposium on Robotics Research*, volume 1. Tokyo, 1989.

[41] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382, Jun 1991.

[42] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 1442–1447 vol.3, Nov 1991.

[43] M. W M G Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, Jun 2001.

[44] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568. IEEE, October 2006.

[45] S.J. Julier and J.K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238–4243 vol.4, 2001.

[46] Shoudong Huang and G. Dissanayake. Convergence analysis for extended kalman filter based slam. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 412–417, May 2006.

[47] Andrew J. Davison and David W. Murray. Mobile robot localisation using active vision. In Hans Burkhardt and Bernd Neumann, editors, *Computer Vision ECCV98*,

volume 1407, pages 809–825. Springer-Verlag, Berlin/Heidelberg, 1998.

[48] Eric W. Nettleton, Peter W. Gibbens, and Hugh F. Durrant-Whyte. Closed form solutions to the multiple-platform simultaneous localization and map building (slam) problem, 2000.

[49] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

[50] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[51] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[52] A Nuchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 6d slam with an application in autonomous mine mapping. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1998–2003 Vol.2, April 2004.

[53] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: a factored solution to the simultaneous localization and mapping problem. *IN PROCEEDINGS OF THE AAAI NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 593—598, 2002.

[54] Robert Sim, Pantelis Elinas, and Matt Griffin. Vision-based SLAM using the rao-blackwellised particle filter. *IN IJCAI WORKSHOP ON REASONING WITH UNCERTAINTY IN ROBOTICS*, 2005.

[55] D. G Lowe. Object recognition from local scale-invariant features. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999*, volume 2, pages 1150–1157 vol.2. IEEE, 1999.

[56] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-Time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[57] Ethan Eade and Tom Drummond. Scalable monocular SLAM. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 469–476, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[58] Chris Mcglone, Edward Mikhail, Jim Bethel, Chris Mcglone, Edward Mikhail, and Jim Bethel. *Manual of Photogrammetry*. American Society for Photogrammetry and Remote Sensing, 1980.

[59] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[60] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[61] Kok Seng Chong and Lindsay Kleeman. Large scale sonarray mapping using multiple connected local maps. In Alexander Zelinsky, editor, *Field and Service Robotics*,

pages 507–514. Springer London, 1998.

[62] J.J. Leonard and H.J.S. Feder. Decoupled stochastic mapping [for mobile robot amp; auv navigation]. *Oceanic Engineering, IEEE Journal of*, 26(4):561–571, Oct 2001.

[63] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*, volume 2, pages 1899– 1906 vol.2. IEEE, September 2003.

[64] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[65] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. RSLAM: a system for Large-Scale mapping in Constant-Time using stereo. *International Journal of Computer Vision*, 94:198–214, June 2010.

[66] M.A. Paskin. Thin junction tree lters for simultaneous localization and mapping. In *Intl. Joint Conf. on Articial Intelligence (IJCAI)*, 2003.

[67] Frank Dellaert and Michael Kaess. Square root sam. In *Robotics: Science and Systems*, pages 177–184, 2005.

[68] P. Krauthausen, A. Kipp, and F. Dellaert. Exploiting locality in slam by nested dissection. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[69] Kai Ni and F. Dellaert. Multi-level submap based slam using nested dissection. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2558–2565, Oct 2010.

[70] Yair Weiss and William Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, October 2001.

[71] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, December 2008.

[72] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 2011.

[73] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. A sliding window filter for incremental SLAM. In Danica Kragic and Ville Kyrki, editors, *Unifying Perspectives in Computational and Robot Vision*, volume 8, pages 103–112. Springer US, Boston, MA, 2008.

[74] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[75] Mark Cummins and Paul Newman. FAB-MAP: probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6): 647 –665, June 2008.

[76] Ingemar J. Cox and J.J. Leonard. Probabilistic data association for dynamic world

modeling: a multiple hypothesis approach. In *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 1287–1294 vol.2, June 1991.

[77] Ingemar J. Cox and John J. Leonard. Modeling a dynamic environment using a bayesian multiple hypothesis approach. *Artificial Intelligence*, 66(2):311 – 344, 1994.

[78] J. Neira and J. D Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, December 2001.

[79] Dirk Hähnel, Sebastian Thrun, Ben Wegbreit, and Wolfram Burgard. Towards lazy data association in slam. In Paolo Dario and Raja Chatila, editors, *Robotics Research. The Eleventh International Symposium*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 421–431. Springer Berlin Heidelberg, 2005.

[80] Frank Dellaert, Steven M Seitz, Charles E Thorpe, and Sebastian Thrun. Em, mcmc, and chain flipping for structure from motion with unknown correspondence. *Machine learning*, 50(1-2):45–71, 2003.

[81] Niko Sünderhauf and Peter Protzel. Switchable constraints vs. max-mixture models vs. rrr - a comparison of three approaches to robust pose graph slam. In *ICRA*, pages 5198–5203, 2013.

[82] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 03 1964.

[83] Rainer Kuemmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[84] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. Tracking multiple moving objects with a mobile robot. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, page 371, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[85] D. Hähnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002*, volume 1, pages 496– 501 vol.1. IEEE, 2002.

[86] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, 22(2):99–116, 2003.

[87] D. Wolf and G. S Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 2, pages 1301– 1307 Vol.2. IEEE, May 2004.

[88] Chieh-Chih Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas. In *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*, volume 1, pages 842– 849 vol.1. IEEE, September 2003.

[89] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in

dynamic environments. In *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*, volume 2, pages 1557– 1563 vol.2. IEEE, September 2003.

[90] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc Van Gool. Dynamic 3D scene analysis from a moving vehicle. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, pages 1–8, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[91] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

[92] Sameer Agarwal, Noah Snavely, StevenM. Seitz, and Richard Szeliski. Bundle adjustment in the large. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision ECCV 2010*, volume 6312 of *Lecture Notes in Computer Science*, pages 29–42. Springer Berlin Heidelberg, 2010.

[93] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):22:1–22:14, October 2008.

[94] Daphne Kollar and Nir Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

[95] Philip Dawid, Steffen L Lauritzen, and David J Spiegelhalter. *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer, 2007.

[96] Andrs Cano and Serafn Moral. Heuristic algorithms for the triangulation of graphs. In Bernadette Bouchon-Meunier, RonaldR. Yager, and LotfiA. Zadeh, editors, *Advances in Intelligent Computing IPMU '94*, volume 945 of *Lecture Notes in Computer Science*, pages 98–107. Springer Berlin Heidelberg, 1995.

[97] Chang-Jin Kim. Dynamic linear models with markov-switching. *Journal of Econometrics*, 60(12):1 – 22, 1994.

[98] C. Bregler. Learning and recognizing human dynamics in video sequences. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 568–574, Jun 1997.

[99] V. Pavlovic, J.M. Rehg, Tat-Jen Cham, and K.P. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 94–101 vol.1, 1999.

[100] B. North, A. Blake, M. Isard, and J. Rittscher. Learning and classification of complex dynamics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(9):1016–1034, Sep 2000.

[101] Sang Min Oh, James M. Rehg, Tucker Balch, and Frank Dellaert. Data-driven mcmc for learning and inference in switching linear dynamic systems. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 944–949. AAAI Press, 2005.

[102] A. V I Rosti and M.J.F. Gales. Rao-blackwellised gibbs sampling for switching linear dynamical systems. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 1, pages I–809–12 vol.1, May

2004.

[103] Zoubin Ghahramani and Geoffrey E. Hinton. Variational Learning for Switching State-Space Models. *Neural Computation*, 12(4):831–864, 2000. doi: 10.1162/089976600300015619.

[104] Uri Lerner and Ronald Parr. Inference in hybrid networks: Theoretical limits and practical algorithms. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 310–318. Morgan Kaufmann Publishers Inc., 2001.

[105] Steffen L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420): 1098–1108, 1992.

[106] Erik B. Sudderth, Alexander T. Ihler, Michael Isard, William T. Freeman, and Alan S. Willsky. Nonparametric belief propagation. *Commun. ACM*, 53(10):95–103, October 2010.

[107] V. Pavlovic and J.M. Rehg. Impact of dynamic model learning on classification of human motion. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 788–795 vol.1, 2000.

[108] K.P. Murphy. Switching kalman filters. Technical report, Citeseer, 1998.

[109] Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, pages 216–223, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[110] Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[111] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 33–42, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[112] Yaakov Bar-Shalom. *Multitarget-multisensor tracking : principles and techniques*. Yaakov Bar-Shalom, Storrs CT, 1995.

[113] James D. Hamilton. Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45(12):39 – 70, 1990.

[114] Vladimir Pavlovic, James M Rehg, and John MacCormick. Learning switching linear models of human motion. In *NIPS*, pages 981–987, 2000.

[115] Sang Min Oh, Ananth Ranganathan, James M Rehg, and Frank Dellaert. A variational inference method for switching linear dynamic systems. 2005.

[116] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[117] Matthew James Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London, 2003.

[118] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, page 886893. Ieee, 2005.

[119] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, page 878885. Ieee, 2005.

[120] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, page 18. IEEE, 2008.

[121] V. Prisacariu and I. Reid. fastHOG - a real-time GPU implementation of HOG. *Department of Engineering Science, Oxford University, Tech. Rep*, 2310(09), 2009.

[122] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8. IEEE, June 2008.

[123] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, page 623630. IEEE, 2010.

[124] J. Berclaz, F. Fleuret, and P. Fua. Robust people tracking with global trajectory optimization. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 744– 750. IEEE, June 2006.

[125] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera People Tracking with a Probabilistic Occupancy Map. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):267282, 2008.

[126] Anton Andriyenko and Konrad Schindler. Globally optimal multi-target tracking on a hexagonal lattice. In *Computer Vision–ECCV 2010*, pages 466–479. Springer, 2010.

[127] J. Berclaz, F. Fleuret, and P. Fua. Multiple object tracking using flow linear programming. In *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*, page 18. IEEE, 2009.

[128] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33 (9):1806–1819, Sept 2011.

[129] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua. Tracking multiple people under global appearance constraints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 137–144, Nov 2011.

[130] J.K. Wolf, A.M. Viterbi, and G.S. Dixon. Finding the best set of K paths through a trellis with application to multitarget tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 25(2):287296, 1989.

[131] Hao Jiang, S. Fels, and J.J. Little. A Linear Programming Approach for Multiple Object Tracking. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, page 18, 2007.

[132] L. Leal-Taixe, G. Pons-Moll, and B. Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker.

In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, page 120127, 2011.

[133] W. Brendel, M. Amer, and S. Todorovic. Multiobject tracking as maximum weight independent set. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, page 12731280, 2011.

[134] B. Leibe, K. Schindler, and L. Van Gool. Coupled detection and trajectory estimation for multi-object tracking. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, page 18. IEEE, 2007.

[135] B. Leibe, K. Schindler, N. Cornelis, and L. Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(10):16831698, 2008.

[136] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. Robust multiperson tracking from a mobile platform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31 (10):18311846, 2009.

[137] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, page 1732, 2004.

[138] A. Andriyenko and K. Schindler. Multi-target tracking by continuous energy minimization. In *CVPR*, 2011.

[139] A. Andriyenko, K. Schindler, and S. Roth. Discrete-continuous optimization for multi-target tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1926–1933, 2012.

[140] J. Vermaak, S.J. Godsill, and P. Perez. Monte carlo filtering for multi target tracking and data association. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(1): 309332, 2005.

[141] Arnaud Doucet, Ba-Ngu Vo, Christophe Andrieu, and Manuel Davy. Particle filtering for multi-target tracking and sensor management. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 1, pages 474–481 vol.1, 2002.

[142] Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[143] J. Vermaak, Arnaud Doucet, and P. Perez. Maintaining multimodality through mixture tracking. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1110–1116 vol.2, Oct 2003.

[144] J. Giebel, D. Gavrila, and C. Schnrr. A Bayesian Framework for Multi-cue 3D Object Tracking. *Computer Vision-ECCV 2004*, 3024:241252, 2004.

[145] Zia Khan, Tucker Balch, and Frank Dellaert. An MCMC-Based particle filter for tracking multiple interacting targets. In Toms Pajdla and Ji Matas, editors, *Computer Vision - ECCV 2004*, volume 3024, pages 279–290. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[146] Zia Khan, Tucker Balch, and Frank Dellaert. MCMC-Based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1805–1918, 2005.

[147] Peter J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711 –732, December 1995.

[148] Yizheng Cai, Nando Freitas, and JamesJ. Little. Robust Visual Tracking for Multiple Targets. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3954 of *Lecture Notes in Computer Science*, page 107118. Springer Berlin Heidelberg, 2006.

[149] M.D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, page 15151522. Ieee, 2009.

[150] M.D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(9):18201833, 2011.

[151] B.N. Vo, S. Singh, and A. Doucet. Sequential Monte Carlo methods for multitarget filtering with random finite sets. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(4):12241245, 2005.

[152] E. Maggio, M. Taj, and A. Cavallaro. Efficient multitarget visual tracking using random finite sets. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(8): 10161027, 2008.

[153] B. Benfold and I. Reid. Stable multi-target tracking in real-time surveillance video. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3457–3464. IEEE, June 2011.

[154] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ., 1991.

[155] J. Ferryman, A. Shahrokni, et al. An overview of the PETS 2009 challenge. 2009.

[156] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the CLEAR MOT metrics. *Journal on Image and Video Processing*, 2008:1, 2008.

[157] A.R. Zamir, A. Dehghan, and M. Shah. GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs. ECCV, 2012.

[158] Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research*, 29(8):958–980, 2010.

[159] César Cadena, Dorian Gálvez-López, Juan D Tardós, and José Neira. Robust place recognition with stereo sequences. *Robotics, IEEE Transactions on*, 28(4):871–885, 2012.

[160] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. *International Journal of Robotics Research*, 32(7):826–840, July 2013.

[161] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

[162] Niko Sünderhauf and Peter Protzel. Switchable constraints for robust pose graph slam. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1879–1884. IEEE, 2012.

[163] P. Pinies, L.M. Paz, S. Haner, and A. Heyden. Decomposable bundle adjustment using a junction tree. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1246–1253, May 2012.

[164] Yasir Latif, Csar Cadena, and Jos Neira. Robust loop closing over time for pose graph slam. *The International Journal of Robotics Research*, 2013.

[165] P. Agarwal, G.D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 62–69, May 2013.

[166] Gim Hee Lee, F. Fraundorfer, and M. Pollefeys. Robust pose-graph loop-closures with expectation-maximization. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 556–563, Nov 2013.

[167] Paul Newman and Kin Ho. Slam-loop closing with visually salient features. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 635–642. IEEE, 2005.

[168] Adrien Angeli, David Filliat, Stéphane Doncieux, and J-A Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *Robotics, IEEE Transactions on*, 24(5):1027–1037, 2008.

[169] Patrick R Amestoy, Timothy A Davis, and Iain S Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4): 886–905, 1996.

[170] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288, 1986.

[171] Aleksandr V Segal and Ian Reid. Latent data association: Bayesian model selection for multi-target tracking. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2904–2911. IEEE, 2013.

[172] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, page 33543361. IEEE, 2012.

[173] Dorian Galvez-Lopez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[174] Simone Ceriani, Giulio Fontana, Alessandro Giusti, Daniele Marzorati, Matteo Matteucci, Davide Migliore, Davide Rizzi, Domenico G Sorrenti, and Pierluigi Taddei. Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27(4):353–371, 2009.

[175] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, Sept 2010.

[176] H. Grimmett, R. Paul, R. Triebel, and I. Posner. Knowing when we don't know: Introspective classification for mission-critical decision making. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4531–4538, May 2013.