Predictive Parameter Estimation for Bayesian Filtering

by

William Vega-Brown

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author . . . .
Department of Mechanical Engineering
June 2, 2013

Certified by . . . .
Nicholas Roy
Associate Professor
Thesis Supervisor

Certified by . . . .
John Leonard
Professor
Thesis Supervisor

Accepted by . . . .
David E. Hardt
Chairman, Committee on Graduate Students

Predictive Parameter Estimation for Bayesian Filtering

by

William Vega-Brown

Submitted to the Department of Mechanical Engineering
on June 2, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

In this thesis, I develop CELLO, an algorithm for predicting the covariances of any Gaussian model used to account for uncertainty in a complex system. The primary motivation for this work is state estimation; often, complex raw sensor measurements are processed into low dimensional observations of a vehicle state. I argue that the covariance of these observations can be well-modelled as a function of the raw sensor measurement, and provide a method to learn this function from data. This method is computationally cheap, asymptotically correct, easy to extend to new sensors, and noninvasive, in the sense that it augments, rather than disrupts, existing filtering algorithms. I additionally present two important variants; first, I extend CELLO to learn even when ground truth vehicle states are unavailable; and second, I present an equivalent Bayesian algorithm. I then use CELLO to learn covariance models for several systems, including a laser scan-matcher, an optical flow system, and a visual odometry system. I show that filtering using covariances predicted by CELLO can quantitatively improve estimator accuracy and consistency, both relative to a fixed covariance model and relative to carefully tuned domain-specific covariance models.

Thesis Supervisor: Nicholas Roy
Title: Associate Professor

Thesis Supervisor: John Leonard
Title: Professor

Acknowledgments

This work would not have been possible without the help of many people.

I thank my advisor, Nick Roy. His insight has been invaluable; his guidance helped shape both the ideas in this thesis, and the way I approach research.

I thank John Leonard, for his willingness to guide me through the hidden paths of my department, as well as for his help and support in all my years at MIT.

I thank my colleagues and fellow students: Abe Bachrach, who first directed me to the problem of covariance prediction; Adam Bry, who pointed me towards a solution; Jonathan Kelly, who helped shape my early design decisions; Charlie Richter, who consistently provided a late-night sounding board for ideas; and Javier Velez and Josh Joseph, who tolerated my obscure measure-theory questions when I needed help or distraction.

I thank my grandmother, for her many years as an inspiration to me, and for her quiet help in making my education possible.

Finally, I thank my parents, without whom I could be neither where I am nor who I am. I am eternally grateful for all you have done for me.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

$\boldsymbol{\Omega}$ An unobserved random variable representing the state of the environment, and more generally, all unknown quantities we are not interested in.

$\mathbf{x}_t$ Vehicle state at time t, or, more generally, latent variables to be estimated.

$\mathbf{K}_t$ The Kalman gain at time t. For a linear Gaussian process model and a linear Gaussian observation model, the Kalman gain allows for the calculation of the optimal posterior state distribution.

$\boldsymbol{\zeta}_t$ Raw sensor measurement at time t, or, more generally, any observed variables used for estimation. This may be very high-dimensional; if the sensor is a camera, $\boldsymbol{\zeta}$ would be the set of pixel values.

$\mathbf{z}_t$ Observation at time t, computed deterministically from the corresponding measurement $\boldsymbol{\zeta}_t$. This will typically be low-dimensional, and ideally will be chosen such that $\mathbf{x}$ is conditionally independent of $\boldsymbol{\zeta}$ given $\mathbf{z}$. For instance, the transform between sequential images is a low-dimensional function of the state and is independent of scene.

$\mathbf{R}_t$ Covariance of the observation noise at time t

$R(\cdot)$ A function mapping raw measurements $\boldsymbol{\zeta}$ to symmetric positive definite observation covariances $\mathbf{R}$.

$h(\mathbf{x})$ A deterministic function defining a nominal observation as a function of the filter state; we represent an observation as $\mathbf{z} = h(\mathbf{x}) + \mathbf{v}$, with $\mathbf{v}$ a vector of *observation noise*. Commonly this function will be linear, or a projection;

however, it may be arbitrarily nonlinear or nonsmooth. This function maps a state in $\mathbb{R}N_{\mathbf{x}}$ to an observation in $\mathbb{R}N_{\mathbf{z}}$.

$\mathbf{H}_t$ In the case where $h\left(\mathbf{x}\right)$ is linear, or can be approximated as linear, we represent the $h\left(\mathbf{x}\right)$ as a possibly time-variant matrix $\mathbf{H}$: $h\left(\mathbf{x}_t\right) = \mathbf{H}_t\mathbf{x}_t$

$\mathbf{v}_t$ Observation noise at time t, defined as the difference between the observation and the nominal observation, $\mathbf{v} = h\left(\mathbf{x}\right) - \mathbf{z}$

$\boldsymbol{\phi}_t$ A vector of *predictor features* used to predict covariances. This vector is composed of functions of the measurements $\boldsymbol{\zeta}$

$\mathbf{Q}_t$ Covariance of the process noise at time t

$f\left(\mathbf{x}\right)$ A deterministic *process function* which maps the state at time t to the nominal state at time $t+1$: $\mathbf{x}_{t+1} = f\left(\mathbf{x}_t\right) + \mathbf{w}_t$, with $\mathbf{w}_t$ a vector of *process noise*. This implies discrete time; for the case of continuous time, we may integrate continuous differential dynamics between discrete steps and represent this integral as the function $f\left(\mathbf{x}\right)$.

$\mathbf{F}_t$ In the case where $f\left(\mathbf{x}\right)$ is linear, or can be approximated as linear, we represent the $f\left(\mathbf{x}\right)$ as a possibly time-variant matrix $\mathbf{F}$: $f\left(\mathbf{x}_t\right) = \mathbf{F}_t\mathbf{x}_t$

$\mathbf{w}_t$ Process noise at time t, defined as the difference between the predicted state $f\left(\mathbf{x}_t\right)$ and the true state $\mathbf{x}t+1$, $\mathbf{w}_t = f\left(\mathbf{x}_t\right) - \mathbf{x}t + 1$

$\mathbf{L}_t$ The Rauch-Tung-Striebel gain at time t. The RTS gain is used in smoothing, much the way the Kalman gain is used in filtering.

$\boldsymbol{\Sigma}_t$ Covariance of the state distribution at time t

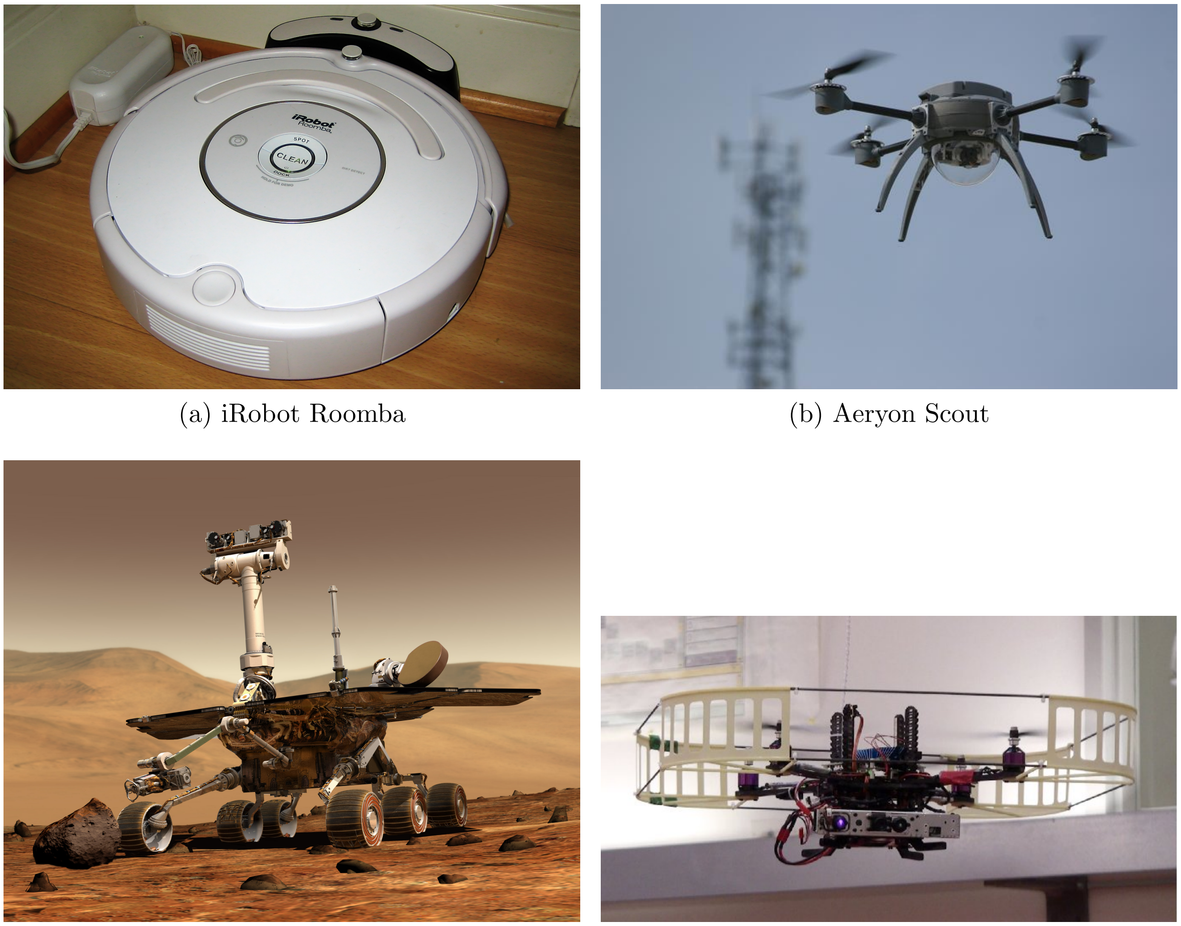
$\boldsymbol{\mu}_t$ Mean of the state distribution at time t

Chapter 1

Introduction

After years of research, autonomous mobile robots are coming of age. Vehicles are increasingly being trusted to navigate and make decisions on their own; this has allowed robots to perform increasingly complex tasks, from automatically cleaning rooms (figure 1-1a), to surveillance and reconnaissance (figure 1-1b), to exploration of other planets (figure 1-1c). These mobile robots are often constrained to have limited sensor and computational payloads, and therefore must operate without perfect information about their state and about the surrounding environment. In order to perform robustly, a robot must be able to reason and plan without perfect information. This is particularly important for small aerial vehicles, like the quadrotor helicopter in figure 1-1d; these vehicles often fly indoors and in cluttered areas, where poor control decisions can lead to catastrophic consequences like collision.

Probabilistic methods have emerged as the dominant way to make decisions in the presence of uncertainty. By explicitly reasoning about distributions over multiple hypotheses, robots are better equipped to make intelligent planning and control decisions in the face of limited information; Thrun et al. [41] argue this point persuasively. Unfortunately, reasoning about uncertainty can carry a considerable computational cost. Care must be taken to ensure that inference is tractable using the computational resources available.



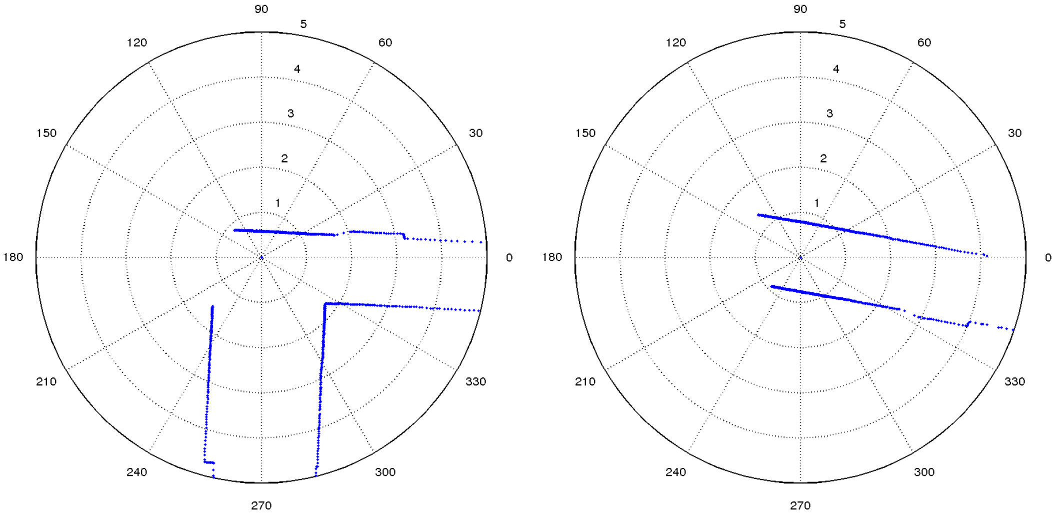
(a) iRobot Roomba

(b) Aeryon Scout

(c) Mars Exploration Rover

(d) Ascending Technologies Pelican

Figure 1-1: *Micro air vehicles like this one require accurate state estimates to fly safely, but are limited in the sensing and computational resources they can carry.*



(a) Corner

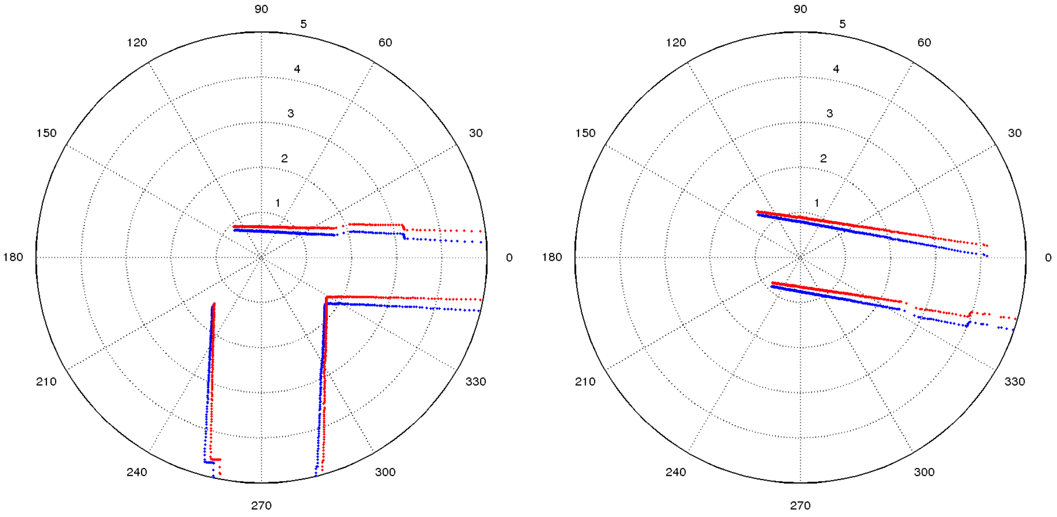
(b) Hallway

Figure 1-2: *The measurements returned by a planar laser scanner depend both on the environment and on the location of the vehicle within that environment.*

1.1 A motivating example

Much research has been done into possible choices of model and representation, and many methods for inference have been presented. However, there exist domains for which all existing solutions are unsatisfying. Consider an aerial vehicle flying indoors, and constrained to use on-board sensing. Such a vehicle cannot directly measure its position using information from GPS, but must instead indirectly measure position using sensors like cameras or LIDAR units, sensors which return measurements that are strongly coupled to both the vehicle state and the surrounding environment. The sample scans from a planar laser scanner presented in figure 1-2 illustrate this coupling; the ranges detected depend on the structure of the surrounding environment.

Because the sensor measurements depend on the structure of the environment, ambiguous structure can lead to uncertainty in the vehicle state, even for a known map and accurate sensors. This is a common phenomenon; consider a human driver travelling on a highway. It is easy for a human to identify in which lane a vehicle is driving, but in the absence of street signs it can be very difficult to determine how far



(a) Corner

(b) Hallway

Figure 1-3: *Ambiguity in the environment can lead to uncertainty. Each image overlays the measurement of a planar laser scanner before and after a small translation. Near a corner, the motion is unambiguous; in a corridor, the sensor information can detect a rotation and a motion across the hallway, but cannot detect motion parallel to the hallway walls.*

along a road the vehicle has driven. Without access to GPS or a similar system, the driver will gradually become more and more uncertain about the absolute position of the vehicle. Similarly, in a corridor, a laser can give information about the location relative to the walls, but if the ends of the corridor are out of range a laser scan gives no information about position along the axis of the hallway. This ambiguity is demonstrated in figure 1-3.

If the environment is known, simple sensor models can capture this uncertainty; the probability $p(s|\mathbf{x}, \mathbf{\Omega})$ of a scan s given a vehicle pose $\mathbf{x}$ and a map $\mathbf{\Omega}$ can be easily computed. Inverting this relation to infer a distribution of a pose given a scan is a well-studied problem. However, if the environment is unknown, the problem is more difficult. One option is to infer a distribution over maps; this is the problem of simultaneous localization and mapping, or SLAM. Solving this problem can be computationally expensive, and if the robot does not require a map to achieve its objectives, SLAM processes are often an undue computational burden.

An alternative approach is to process the measurements in an attempt to decouple the sensor measurements from the unknown environment. For example, aligning sequential laser scans gives a displacement between frames which is largely independent of the structure of the environment; this displacement is an indirect observation of the vehicle velocity. Often we can model the distribution over such a processed measurement as multivariate Gaussian random variable—that is, as a single hypothesis with an associated uncertainty.

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{H}\mathbf{x}, \mathbf{R}) \tag{1.1}$$

Here, the matrix $\mathbf{H}$ is a projection from the state space to the observation space, while $\mathbf{R}$ is a covariance matrix parameterizing the uncertainty associated with the observation.

This model is convenient; it enables exact, efficient inference using the Kalman filter [25]; I review this algorithm, and several important variants, in chapter 2. However, choosing this model introduces a new problem: identifying the covariance $\mathbf{R}$. Choosing a single fixed covariance cannot model variations in uncertainty due to environmental ambiguity. Fixed covariance models can be, and have been, employed successfully, but choosing the parameters appropriately is a matter of touch and tuning, and requires trading off the cost of uncertainty associated with an inflated covariance against the risk of inconsistency or incorrectness if the covariance is set too small. Predicting observation covariances, and explicitly allowing that covariance to vary, will lead to strictly superior performance.

1.2 Covariance prediction

This thesis is concerned with the task of predicting the uncertainty of a observation computed from a raw measurement. The problem of covariance prediction is not limited to state estimation; predictive covariance schemes can be useful any time a variable is computationally abstracted from raw sensor data and treated as a Gaussian. This occurs in constraint based SLAM systems [30], in speech recognition

[16, 38], in financial time series forecasting [19], and elsewhere. However, I focus on state estimation as a concrete example of the utility of predicting covariances.

Covariance prediction is not a new problem, but there is no consensus on how predictions ought to be made, nor even on what the covariance should depend. Commonly, the covariance is assumed to be time-varying, or state-dependent; I argue that the covariance instead should be predicted based on the sensor measurement from which the observation was computed. This has the advantage of flexibility, while still promising generality: it is reasonable to assume, for example, that any corridor-like environment will provide information about just one direction, regardless of whether that corridor is in an office in Cambridge or a cave on Mars. By formulating the prediction problem so that the covariance depends on the sensor measurement, the predictions are shown to generalize to new environments without access to training data in those environments.

Having formulated the prediction problem, I then develop an algorithm for learning a covariance model by maximizing the likelihood of the observation model. The learning objective gives the algorithm its name: Covariance estimation through learned likelihood optimization, or CELLO. This algorithm is shown to be computationally efficient and asymptotically correct, both under a frequentist and a Bayesian interpretation of statistics. In addition, I employ expectation maximization to generalize CELLO to the unsupervised case, where ground truth estimates of state are unavailable; the modified algorithm, called CELLO-EM, retains the asymptotic correctness of the supervised variation.

Finally, I describe the details of my implementation of CELLO, and provide a guide for usage. Using this implementation, I validate the algorithm, both in simulation and on real data taken from several sensors, including a planar laser scanner, a monochrome camera, and an RGB-D sensors. I show that using a dynamic covariance model can lead to dramatic improvements over a fixed covariance model, and additionally show that CELLO outperforms competing state-of-the-art algorithms, even in the domains for which those algorithms are designed and tuned.

Chapter 2

Recursive Bayesian estimation

Recall the motivating example of from the previous chapter. A robot navigates an unknown environment, and at discrete intervals receives raw sensor measurements $\boldsymbol{\zeta}_t \in \mathbb{R}^{N_\zeta}$, where $t \in \mathbb{Z}$ is the index of the interval in which a measurement is received. Throughout this thesis, the dimensionality of any vector $\mathbf{v}$ will be denoted as $N_{\mathbf{v}}$; $N_{\boldsymbol{\zeta}}$ is then the dimensionality of the measurement vector $\boldsymbol{\zeta}$. Based on the information those sensor measurements provide about the state of the vehicle and the state of the world, the robot can generate actuator commands to accomplish whatever task has been assigned to it.

It is often simpler to first calculate an estimate of the vehicle state, and then make decisions based on that estimated state[1]. Decoupling the planning and control problem from the sensor processing problem can lead to more efficient, more interpretable, and better-performing planners and controllers. In particular, the decoupling enables planning at a higher level: it is difficult to express a command like “Move forwards ten meters” in terms of a camera image, but it is trivial to express in terms of a position estimate.

I represent the environment by the random vector $\boldsymbol{\Omega}$, and denote the vehicle state at time[2] t as $\mathbf{x}_t$, drawn from a state space $\mathcal{X}$. For a rigid body in three dimensions,

[1] It is possible to make control decisions without a state estimate, or indeed without a model of the state at all. An example of this approach is the predictive state representation introduced by Littman et al. [29]. I restrict my discussion to problems with a clear state to be estimated, and do not further consider such approaches.

[2] The representation of time as discrete is necessary for implementation on a digital computer,

the state would include a three-dimensional position vector and three orientation parameters, as well as the first derivatives of these parameters, for a total of twelve elements; therefore, for a rigid body $\mathcal{X} = \mathbb{R}^{12}$. The state at any time step is assumed to have the Markov properly: it is conditionally dependent only upon the state at the previous time step.

$$p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots, \mathbf{x}_0) = p(\mathbf{x}_t | \mathbf{x}_{t-1}) \tag{2.1}$$

The distribution[3] $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ I refer to as the *process model.* This distribution encapsulates both how the state will evolve, and how uncertainty will accumulate as our predictions extend into the future. Formally, the state sequence $\{\mathbf{x}_t \quad \forall t \in \mathbb{Z} \geq 0\}$ constitutes a discrete-time Markov process.

Given an initial distribution $p(\mathbf{x}_0)$ it is straightforward to compute the distribution at any future time by marginalizing the intervening states. There are a number of approaches to inferring a state sequence $\mathbf{x}_{1:N} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, based on a sequence of available measurements $\boldsymbol{\zeta}_{1:N} = \{\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_N\}$. In this chapter, I present a review of common techniques and approximations employed to solve the state estimation problem. I attempt to illustrate that, for some domains, all existing methods are unsatisfying, and I argue for an alternate model to be employed instead.

2.1 Hidden Markov models

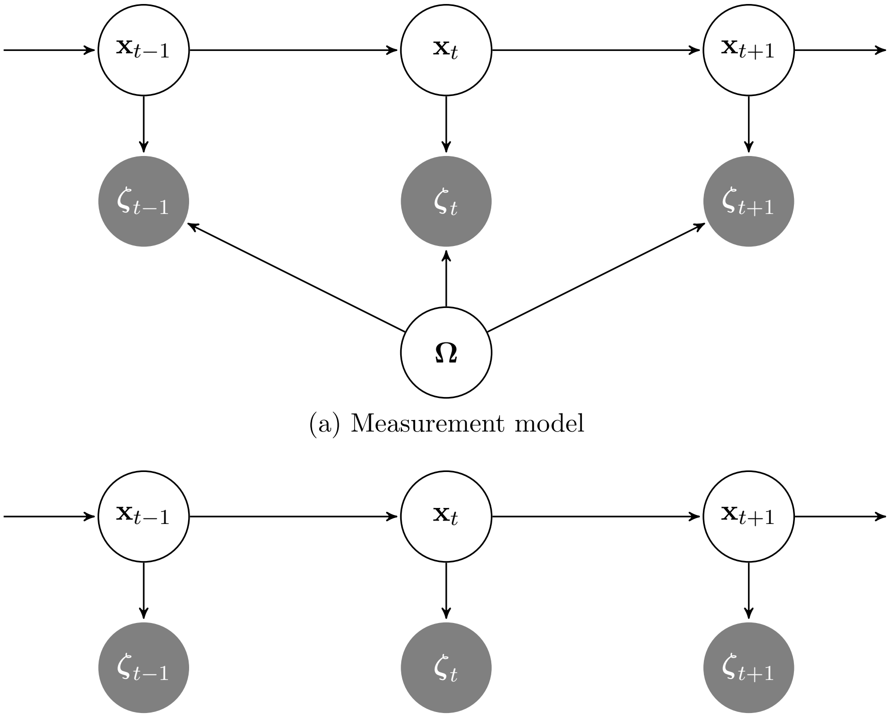
Sensors generate data by interacting with the environment; as such, our sensor measurements depend both on the vehicle state $\mathbf{x}$ and on the environment $\boldsymbol{\Omega}$. The de-

but does not represent a significant loss of generality; the state of a continuous-time system at any countable collection of times may be represented without approximation as a discrete-time system.

[3] As a mathematical aside, when I refer to a distribution $p(\mathbf{x})$ over a space $\mathcal{X}$, I mean a marginal density with respect to a dominating measure. I abuse notation and refer to this measure as d$\mathbf{x}$; therefore, the expression

$$\int_{\mathcal{X}} \mathrm{d}\mathbf{x}\, p(\mathbf{x})\, f(\mathbf{x}) \tag{2.2}$$

should be interpreted as a Lebesgue integral, and give the expectation $\mathbf{E}[f]$ regardless of whether $\mathcal{X}$ is discrete or continuous. In the case where $\mathcal{X} \subseteq \mathbb{R}^n$, the integral of equation (2.2) may be evaluated as a multidimensional Riemann integral; if $\mathcal{X} \subseteq \mathbb{Z}^n$, the integral should be evaluated as a sum.



(a) Measurement model

(b) Hidden Markov model

Figure 2-1: *The true measurement model in figure 2-1a reduces to the hidden Markov model of figure 2-1b if the environment is observed or if the measurements are independent of the environment. The hidden Markov model has the important property that the state $\mathbf{x}_t$ as time t depends only on the previous state $\mathbf{x}_{t-1}$ and the current measurement $\boldsymbol{\zeta}_t$.*

pendency structure is depicted graphically in figure 2-1a. If the environment is observable, or if the measurements are independent of the environment, this structure can be reduced to the familiar form of a hidden Markov model, as in figure 2-1b.

When all relevant environmental parameters are known, or if our sensor is such that measurements are independent of the environment, the system has the dependency structure of a hidden Markov model. For example, a ground vehicle using wheel odometry to estimate position takes measurements which depend on vehicle velocity but are nominally independent of its environment. In the case of the quadrotor helicopter described in chapter 1, if the vehicle is navigating in a known map, sequential scans taken by a laser range finder are independent, and depend only on the vehicle pose.

I refer to an *observation* $\mathbf{z}_t \in \mathcal{Z}$ of the state $\mathbf{x}_t$ at time step t as distinct from a measurement $\boldsymbol{\zeta}_t$. A measurement refers specifically to raw data collected by a sensor; by construction, an observation is statistically independent of all other observations as well as the vehicle state at all other time steps.

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_t, \mathbf{z}_{t-1}, \dots, \mathbf{z}_0) = p(\mathbf{z}_t|\mathbf{x}_t) \tag{2.3}$$

It follows that a measurement $\boldsymbol{\zeta}$ may or may not constitute an observation of the vehicle state $\mathbf{x}$, depending on whether it depends on the environment $\boldsymbol{\Omega}$. The set $\mathcal{Z}$ denotes a space of possible observations, and may be continuous or discrete. I refer to the distribution $p(\mathbf{z}_t|\mathbf{x}_t)$ as the *observation distribution*, though it is referred to in some works as the emission distribution or measurement distribution.

There are several queries to make of the model defined by the process and observation distributions. First, we may seek to infer a distribution over the state $\mathbf{x}_t$ from the available sequence of past observations $\mathbf{z}_{0:t} = \{\mathbf{z}_0, \dots, \mathbf{z}_t\}$. This is the *filtering problem*, and has an exact, recursive solution, comprised of two subroutines are called in sequence at each time step. The *prediction step* evaluates the distribution over the state $\mathbf{x}_t$, given all previous observations—that is, it predicts the current state given the state estimate at the previous time step.

$$p(\mathbf{x}_t|\mathbf{z}_{0:t-1}) = \int \mathrm{d}\mathbf{x}_{t-1} p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{z}_{0:t-1}) \tag{2.4}$$

Next, the *update step* evaluates

$$p(\mathbf{x}_t|\mathbf{z}_{0:t}) \propto p(\mathbf{z}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{z}_{0:t-1}) \tag{2.5}$$

By virtue of the Markov condition and the independence of the observations, we only ever need to keep track of a single time-varying distribution, $p(\mathbf{x}_t|\mathbf{z}_{0:t})$, in order to infer the current state given the entire history of observations.

We may also be interested in obtaining the distribution over past states $\mathbf{x}_{0:t}$, given all measurements up to the present; that is, evaluating $p(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})$. This is the *smoothing problem.* The Markov condition implies the independence of past and future measurements at any time t:

$$
\begin{aligned}
p(\mathbf{x}_t|\mathbf{z}_{0:T}) &= p(\mathbf{x}_t|\mathbf{z}_{0:t})\, p(\mathbf{x}_t|\mathbf{z}_{t+1:T}) && (2.6)\\
&= p(\mathbf{x}_t|\mathbf{z}_{0:t})\, p(\mathbf{x}_t|\mathbf{x}_{t+1})\, p(\mathbf{x}_{t+1}|\mathbf{z}_{0:T}) && (2.7)
\end{aligned}
$$

By first running the forward algorithm, we may evaluate the first term of equation (2.7) for each t. The second term is simply the time-reversed dynamics of the system; for many classes of models, this is trivial to obtain. The third term may be evaluated recursively; the last state in the sequence, $\mathbf{x}_T$, has $p(\mathbf{x}_T|\mathbf{z}_{0:T})$ given by the forward algorithm. The preceding state $p(\mathbf{x}_{T-1}|\mathbf{z}_{0:T})$ may be evaluated recursively from equation (2.7), and so on. We may obtain $p(\mathbf{x}_{0:t}|\mathbf{z}_{0:t})$ by first running the forward algorithm, and then running a similar procedure backwards; accordingly, the full procedure is known as the *forward-backward algorithm.*

For simplicity, I have assumed the process and observation distributions do not vary with time. It is easy to extend to the time-varying case; in fact, the forward and forward-backward algorithms are completely general, provided we know the initial state distribution $p(\mathbf{x}_0)$, the process model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ at each time step t, and the observation model $p(\mathbf{z}_t|\mathbf{x}_t)$ at each time step t.

Implementing the forward or the forward-backward algorithm requires evaluating the integrals and products in equations (2.4), (2.5) and (2.7). Unfortunately, exact evaluation is possible only for a very limited set of models. For other models, we must approximate; such approximations are well studied, and represent the state of the art of estimation.

Note that we may always induce the structure of a hidden Markov model given the augmented Markov model that describes the state estimation problem. Informally, if we infer both the vehicle state $\mathbf{x}$ and the environmental parameters $\boldsymbol{\Omega}$, the measurements satisfy the requisite conditions. This is equivalent to running the junction-tree algorithm on the augmented model, and leads to the simultaneous localization and mapping (SLAM) problem. Substantial work has been done on generating tractable and efficient algorithms for solving this problem; I will not discuss such algorithms in this thesis.

2.1.1 Finite State Space filter

Suppose the state space $\mathcal{X}$ and the observation space $\mathcal{Z}$ are both discrete and finite. This is useful for resolving classification queries; for instance, a robot may behave differently outdoors than indoors, and could represent that state by a binary random variable. I denote $\mathcal{X} = \{1, \ldots, M\}$ and $\mathcal{Z} = \{1, \ldots, N\}$, and represent the probability of a state i at time t as p_i^t:

$$p(\mathbf{x}_t = i) = p_i^t \tag{2.8}$$

The process and observation models can be represented as Markov matrices $\mathbf{F} \in \mathbb{R}^{M\times M}$ and $\mathbf{H} \in \mathbb{R}^{N\times M}$, so that $p(\mathbf{x}_{t+1} = i|\mathbf{x}_t = j) = \mathbf{F}_{ij}$ and $p(\mathbf{z}_t = i|\mathbf{x}_t = j) = \mathbf{H}_{ij}$. In that case, the prediction, update, and smoothing processes can be expressed in closed form.

$$\textbf{Prediction:} \qquad p_i^{t+1|1:t} = \sum_{j=1}^{M} \mathbf{F}_{ij} p_j^{t|1:t} \tag{2.9}$$

$$\textbf{Update:} \qquad p_i^{t|1:t} = \frac{\mathbf{H}_{\mathbf{z}_t i} p_i^{t|1:t-1}}{\sum_{j=1}^{M} \mathbf{H}_{\mathbf{z}_t j} p_j^{t|1:t-1}} \tag{2.10}$$

$$\textbf{Smoothing:} \qquad p_i^{t|1:T} = p_i^{t|1:T} \sum_{j=1}^{M} (\mathbf{F}^{-1})_{ij} p_j^{t|1:T} \tag{2.11}$$

It is more standard for discrete hidden Markov models to evaluate the backward step independently of the forward step. The resulting distribution is identical, however.

The finite state space model is of limited utility in state estimation; typically, the vehicle state is continuous. If we bound the state space and discretize, we obtain a finite space representation; however, the size of this representation scales exponentially with the state dimension, and so this is rarely tractable. Moreover for many vehicles a discretized representation is wasteful: the majority of the probabilistic weight is concentrated among comparatively few discretized states, yet the algorithm devotes equal resources to the likely and unlikely states.

2.1.2 Particle filtering

An alternative to discretizing the state space is the particle filter, first presented by Gordon et al. [18]. The particle filter is a recursive, natively continuous solution to hidden Markov model inference, which can be used with arbitrary process and observation distributions. This method approximates the distribution over the state $\mathbf{x}_t \in \mathbb{R}^{N_\mathbf{x}}$ at time t by a weighted sum of a finite number of samples:

$$p(\mathbf{x}_t) \propto \sum_{i=1}^{t} w_t^i \delta(\mathbf{x}_t - \mathbf{x}_t^i) \tag{2.12}$$

where proportionality becomes equality if the weights sum to one. There are many particle filtering algorithms; one of the simplest is sequential importance sampling.

$$\textbf{Prediction:} \qquad \mathbf{x}_{t+1}^i \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t) \tag{2.13}$$

$$\textbf{Update:} \qquad w_t^i = w_{t-1}^i p\big(\mathbf{z}_t|\mathbf{x}_t^i\big) \tag{2.14}$$

There is no simple smoothing procedure for particle filters.

Implementations of the particle filter typically will resample to concentrate particles in the most likely regions of probability space. Particle filters require only that the process and observation models are known; they can tolerate any kind of distribution, including non-stationary models. They are computationally expensive, however; to ensure good performance for high-dimensional state spaces, the filter requires a large numbers of particles be stored and propagated at each time step.

2.1.3 Kalman filtering

If insufficient computational resources are available to run a particle filter, it may be sufficient to restrict the class of state distributions to some parametric family. When the process and observation models are uncertain but unimodal, they are often well modelled as a deterministic function of the state, corrupted by white noise.

$$\mathbf{x}_{t+1} = f\left(\mathbf{x}_t\right) + \mathbf{w}_t \tag{2.15}$$

$$\mathbf{z}_t = h\left(\mathbf{x}_t\right) + \mathbf{v}_t \tag{2.16}$$

The process noise $\mathbf{w}_t$ and observation noise $\mathbf{v}_t$ are additive zero-mean Gaussian random variables with covariance $\mathbf{Q}_t$ and $\mathbf{R}_t$, respectively.

$$\mathbf{w}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_t\right) \tag{2.17}$$

$$\mathbf{v}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_t\right) \tag{2.18}$$

The Gaussian distribution has several properties that make it mathematically attractive; it is the maximum entropy distribution over the real numbers given a fixed first and second moment; it is closed under conditioning and marginalization; and it is a *stable* distribution. If a variable $\mathbf{x}$ is normally distributed,

$$\mathbf{x} \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \tag{2.19}$$

then it will remain normally distributed after any linear transformation.

$$\mathbf{A}\mathbf{x} + \mathbf{b} \sim \mathcal{N}\left(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top\right) \tag{2.20}$$

These properties imply that if the initial state is distributed as a multivariate Gaussian:

$$p(\mathbf{x}_0) = \mathcal{N}\left(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\right) \tag{2.21}$$

and if the process and observation models are linear and Gaussian:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{F}_t\mathbf{x}_{t-1}, \mathbf{Q}_t\right) \tag{2.22}$$

$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{H}_t\mathbf{x}_t, \mathbf{R}_t\right) \tag{2.23}$$

then the state distribution will remain Gaussian at all future time steps. This is called the *linear Gaussian* model, and for this model the prediction, update, and smoothing steps may be evaluated exactly and in closed form, resulting in the well-known Kalman filter [25].

$$\begin{aligned}
\textbf{Prediction:} \qquad p(\mathbf{x}_t|\mathbf{z}_{0:t-1}) &= \mathcal{N}\left(\boldsymbol{\mu}_{t|0:t-1}, \boldsymbol{\Sigma}_{t|0:t-1}\right) \\
\boldsymbol{\mu}_{t|0:t-1} &= \mathbf{F}_t \boldsymbol{\mu}_{t-1|0:t-1} && (2.24) \\
\boldsymbol{\Sigma}_{t|0:t-1} &= \mathbf{F}_t \boldsymbol{\Sigma}_{t-1|0:t-1} \mathbf{F}_t^\top + \mathbf{Q}_t && (2.25) \\
\textbf{Update:} \qquad p(\mathbf{x}_t|\mathbf{z}_{0:t}) &= \mathcal{N}\left(\boldsymbol{\mu}_{t|0:t}, \boldsymbol{\Sigma}_{t|0:t}\right) \\
\boldsymbol{\mu}_{t|0:t} &= \boldsymbol{\mu}_{t|0:t-1} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}_t \boldsymbol{\mu}_{t|0:t-1}) && (2.26) \\
\boldsymbol{\Sigma}_{t|0:t} &= (\mathbb{1} - \mathbf{K}_t \mathbf{H}_t) \boldsymbol{\Sigma}_{t|0:t-1} && (2.27) \\
\mathbf{K}_t &= \boldsymbol{\Sigma}_{t|0:t-1} \mathbf{H}_t^\top (\mathbf{H}_t \boldsymbol{\Sigma}_{t|0:t-1} \mathbf{H}_t^\top + \mathbf{R}_t)^{-1} && (2.28) \\
\textbf{Smoothing:} \qquad p(\mathbf{x}_t|\mathbf{z}_{0:T}) &= \mathcal{N}\left(\boldsymbol{\mu}_{t|0:T}, \boldsymbol{\Sigma}_{t|0:T}\right) \\
\boldsymbol{\mu}_{t|0:T} &= \boldsymbol{\mu}_{t|0:t} + \mathbf{L}_t(\boldsymbol{\mu}_{t+1|0:T} - \boldsymbol{\mu}_{t+1|0:t}) && (2.29) \\
\boldsymbol{\Sigma}_{t|0:T} &= \boldsymbol{\Sigma}_{t|0:t} + \mathbf{L}_t(\boldsymbol{\Sigma}_{t+1|0:T} - \boldsymbol{\Sigma}_{t+1|0:t}) \mathbf{L}_t^\top && (2.30) \\
\mathbf{L}_t &= \boldsymbol{\Sigma}_{t|0:t} \mathbf{F}_t^\top \boldsymbol{\Sigma}_{t+1|0:t}^{-1} && (2.31)
\end{aligned}$$

These results follow naturally from the definitions of each step and properties of the Gaussian distribution, although the derivations entail substantial algebraic manipulation. The Kalman filter is an exact solution to the filtering problem on hidden Markov models, provided all distributions are Gaussian, all mean functions are linear, and all parameters are known. Moreover, compared to the particle filter or the finite state space filter, the Kalman filter is highly efficient. The most expensive step is the inversion of symmetric covariance matrices; the complexity of this operation scales with a polynomial, rather than an exponential, function of the state space dimension $N_\mathbf{x}$ and the observation space dimension $N_\mathbf{z}$. Because of this polynomial complexity, the Kalman filter is tractable even for very large state spaces.

2.1.4 Non-linear Kalman Filters

The restriction to linear observation and process functions can limit the utility of the Kalman filter. Often, sensors are fixed to a moving robot; to estimate the state in a fixed frame, we must rotate the sensor measurements, which is a non-linear transform. There are many variants on the Kalman filter designed to handle nonlinearities. The oldest and most well-known is the extended Kalman filter, first presented by Gelb [15]. The extended Kalman filter makes a locally linear approximation to nonlinear transition and observation functions. In particular, it approximates

$$h\left(\mathbf{x}_t\right) \approx h\left(\boldsymbol{\mu}_t\right) + \nabla h\left(\boldsymbol{\mu}_t\right)\mathbf{x}_t \tag{2.32}$$

$$f\left(\mathbf{x}_t\right) \approx f\left(\boldsymbol{\mu}_t\right) + \nabla f\left(\boldsymbol{\mu}_t\right)\mathbf{x}_t \tag{2.33}$$

This implies the observation matrix $\mathbf{H}_t = \nabla h\left(\boldsymbol{\mu}_t\right)$ and the transition matrix $\mathbf{F}_t = \nabla f\left(\boldsymbol{\mu}_t\right)$. Provided the covariance is small enough that higher derivatives of the observation and process functions can be ignored, this approximation performs quite well. Other variants, like the unscented Kalman filter [24] or the cubature Kalman filter [3], approximate the posterior differently, and extract different advantages from doing so. It is possible to derive closed form recursive filters for skewed [37] or platykurtic [1] noise distributions. The underlying mathematics is fundamentally similar but mechanically complex, and I do not discuss such variants here.

2.2 Preprocessing measurements

The Kalman filter provides an efficient framework for inference, provided the observation function is smooth and continous. For complex measurements like those returned by a camera or planar laser, measurements which are discontinuous functions of both the state and the environment, the Kalman filter is not immediately applicable. Often, it is possible to process the raw measurements $\boldsymbol{\zeta}$, and generate observations $\mathbf{z}$ that are smooth, low dimensional functions of the vehicle state.

Given a planar laser scan in a known environment, for example, it is possible to localize by particle filtering in the six dimensional space of positions and orientations, rather than the full twelve dimensional state space of the vehicle. This low-dimensional localization can then be incorporated into a Kalman filter as a noisy, but smoothly varying, observation of the vehicle state. This is the approach taken by Bry et al. [12], and results in a significant computational win.

If a map is not available, it is possible to compute a transformation between sequential laser scans, and treat these transformations as observations of the vehicle velocity. This process is known as *odometry*, and can be done with a variety of sensors. If the transformations are computed from laser scans, the process is called *scan-matching*; if they are computed from a sequence of images, the process is *visual odometry*. This provides an efficient way of estimating the full state of a vehicle, even for complex sensors with strong dependency on an unknown environment. In effect, the computation of an odometry observation acts as a high-pass filter, removing the low frequency signal that is the unknown but fixed map and leaving behind just the vehicle state.

2.3 Covariance models

Although processing raw measurements into observations permits the use of efficient hidden Markov model inference without the need to infer a description of the environment, it induces a new problem. The uncertainty of an observation, parameterized by the covariance, must be carefully chosen in order to ensure good performance. If the covariance is chosen poorly, the inferred state distributions may become inconsistent or even divergent.

The problem of choosing the noise parameters in a Kalman filter is not new; it has attracted attention for many years. The most common solution is to simply choose parameters which yield good performance; this is the approach taken in the seminal work of Kalman [25]. Kalman demonstrated his eponymous filter was the least-squares optimal solution to the filtering problem; it solved the same problem as a linear-quadratic regulator, with the noise covariances $\mathbf{R}$ and $\mathbf{Q}$ filling the roles of the positive definite gain matrices defining the quadratic cost function.[4]

In that sense, the noise covariances are simply gain matrices representing the cost of filter error, with a cost function defined as

$$\begin{aligned} C(\mathbf{x}_{0:T}, \mathbf{z}_{0:T}) = & \sum_{t=0}^{T} (\mathbf{z}_t - h\left(\mathbf{x}_t\right))^\top \mathbf{R}_t (\mathbf{z}_t - h\left(\mathbf{x}_t\right)) \\ & + \sum_{t=1}^{T} (\mathbf{x}_t - f\left(\mathbf{x}_{t-1}\right))^\top \mathbf{Q}_t (\mathbf{x}_t - f\left(\mathbf{x}_{t-1}\right)) \end{aligned} \tag{2.34}$$

These gains may be tuned by hand or chosen heuristically. Using the filter in this way is valid, and often results in satisfactory performance. This interpretation also illustrates the phenomena that scaling both $\mathbf{R}$ and $\mathbf{Q}$ by an identical constant factor will not affect the estimated trajectory $\boldsymbol{\mu}_{0:T}$, although it will affect the estimated state covariances. If the covariances are simply being discarded, as is often the case, this overall scale factor may be safely ignored.

However, if we wish to explicitly reason about uncertainty, we require that the state estimate covariance be equal to, or at least an approximation of, the true covariance of the estimate. Reasoning about uncertainty can be crucial for insuring robustness. If, for instance, the standard deviation of the position error is much smaller than the distance to the nearest obstacle, commanding an aerial vehicle to make aggressive maneuvers is sensible. If there are obstacles inside the 2σ covariance ellipse, then with high probability aggressive maneuvers will result in a high-speed collision.

For such situations, the Bayesian interpretation[5] of the filtering problem is essential, as is choosing the noise parameters to accurately reflect the process and observation distributions. Moreover, in the general setting of recursive Bayesian estimation, the distribution parameters may not have a clear interpretation within a

[4] Note that I explained the Kalman filter in the context of recursive Bayesian inference on a linear Gaussian model, while Kalman derived it as the least-squares optimal linear filter regardless of noise distribution. Both interpretations yield the same filter.

[5] Here, Bayesian refers to Bayes' theorem, and not to the Bayesian interpretation of statistics. It is used to contrast the least-squares interpretation; both the frequentist and Bayesian interpretation of statistics agree on the form of a recursive Bayesian estimator, although they would disagree on how to interpret the resulting distributions.

cost function, and so it may not be clear how to make reasonable, let alone optimal, choices. We require a principled, and ideally algorithmic, approach.

2.3.1 Optimal fixed-parameter models

Rather than tune the process and measurement covariances by hand, they may be inferred via the Baum-Welch algorithm [8], which is a special case of the Expectation-Maximization algorithm [14], applied to hidden Markov models. Formally, given N_D observations $\{\mathbf{z}_1, \ldots, \mathbf{z}_{N_D}\}$, along with an initial guess at the parameters $\boldsymbol{\theta}_0$, the Baum-Welch algorithm evaluates

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} p(\mathbf{z}_i|\boldsymbol{\theta}) \tag{2.35}$$

$$= \arg\max_{\boldsymbol{\theta}} \int \mathrm{d}\mathbf{x}_1 \ldots \mathbf{x}_{N_D} \prod_{i=1}^{N} p(\mathbf{x}_i|\mathbf{x}_{i-1})\, p(\mathbf{z}_i|\boldsymbol{\theta}) \tag{2.36}$$

This algorithm evaluates this optimization efficiently by iteratively improving the estimated parameters, in two alternating steps. First, we evaluate the latent variables conditioned on the current parameter estimate:

$$\hat{\mathbf{x}}_1^{(n)} \ldots \hat{\mathbf{x}}_N^{(n)} = \mathbf{E}\left[\mathbf{x}_1 \ldots \mathbf{x}_N|\mathbf{z}_i, \boldsymbol{\theta}_n\right] \tag{2.37}$$

This expectation may be found using the forward-backward algorithm. Then, we evaluate the maximum likelihood parameters given the distribution over the latent states found during the expectation step.

$$\boldsymbol{\theta}^\star_{n+1} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} p\Big(\mathbf{z}_i|\mathbf{x}_i^{(n)}, \boldsymbol{\theta}\Big)\, p\Big(\mathbf{x}_i^{(n)}\Big) \tag{2.38}$$

Iterating between these steps will always converge to a local maxima.

If the parameters to be estimated are the covariances of a fixed-covariance linear Gaussian model, then the expectation step of equation (2.37) reduces to running a Kalman filter forward and smoothing the data after, according to equations equations (2.24), (2.26) and (2.29). The maximization step has a closed form solution:

$$\hat{\mathbf{R}}^{(n)} = \frac{1}{N_D + 1} \sum_{i=1}^{N_D} (\mathbf{z}_i - \mathbf{H}_i \hat{\mathbf{x}}_i^{(n)})(\mathbf{z}_i - \mathbf{H}_i \hat{\mathbf{x}}_i^{(n)})^\top \tag{2.39}$$

$$\hat{\mathbf{Q}}^{(n)} = \frac{1}{N_D + 1} \sum_{i=1}^{N_D} (\hat{\mathbf{x}}_i - \mathbf{F}_{i-1} \hat{\mathbf{x}}_{i-1}^{(n)})(\hat{\mathbf{x}}_i - \mathbf{F}_{i-1} \hat{\mathbf{x}}_{i-1}^{(n)})^\top \tag{2.40}$$

It can be shown that for a fixed-covariance linear Gaussian model this optimization is convex and thus will converge in finite time to the unique global maximum-likelihood parameters.

Unfortunately, the best fixed noise parameters may still give suboptimal results. Recall the example of scan-matching in a hallway; when the laser cannot see the far ends of a straight-walled corridor, the observation computed from the measurements provides no information about motion parallel to the walls. A fixed covariance model cannot capture this behavior.

2.3.2 Adaptive parameter models

Adaptive parameter models assume the parameters are a function of time.

$$\mathbf{R}_t = R(t) \tag{2.41}$$

These models extend the inference procedure to estimate the noise parameters in tandem with the state vector. In the context of the recursive Bayesian framework, they infer

$$p(\mathbf{x}_{0:T}, \mathbf{R}_{0:T}, \mathbf{Q}_{0:T} | \mathbf{z}_{0:T}) \tag{2.42}$$

Alternatively, some adaptive methods choose to minimize some cost function of the sequences of $\mathbf{x}$, $\mathbf{R}$, and $\mathbf{Q}$, such as the weighted squared error of equation (2.34). The earliest example of this approach was given by Mehra [33], called the adaptive Kalman filter. The adaptive Kalman filter conducts a statistical optimality test at each time step, and if the test determines the covariances are suboptimal, the covariances are updated. These updates guarantee asymptotically unbiased and consistent estimates of the covariance matrices. That is, if there exist fixed $\mathbf{R}$ and $\mathbf{Q}$, and if the system is linear Gaussian, then in the limit of infinite samples the state estimate $\mathbf{x}$ will be unbiased and the covariances $\mathbf{R}$ and $\mathbf{Q}$ will converge to their true values.

This filter is most useful if the covariances are known to be fixed, but no data is available before run time for training. Without prior data, expectation-maximization cannot be applied. However, the trifecta of linearity, Gaussianity, and stationarity is rarely achievable in the domains of interest. Even approximate stationarity implies the time scale over which the noise parameters vary is much greater than the time scale over which the state varies. As such, the adaptive Kalman filter can make no performance guarantees in the face of rapid changes in the noise parameters, even if it were possible to guarantee linearity and Gaussianity.

Others have used the same basic strategy, with differing update criteria and methods. For instance, the adaptive fading extended Kalman filter of Kim et al. [26] computes the measurement and process covariances from a sliding window of recent measurements. Hu et al. [22] use a similar strategy, recursively updating the noise parameters as if they were constant but utilizing a forgetting factor to place more emphasis on more recent innovation covariances.

Any adaptive strategy is faced with a fundamental limitation; in order for adaption to function, the noise parameters must be tightly correlated in time. This is certainly the case for fixed or slowly varying parameters, but there is no reason to assume *a priori* that this is the case. In fact, for sufficiently complex sensor processing algorithms, it is often *not* the case; consider the example of laser scan-matching odometry given previously. If a robot in a hallway yaws rapidly back and forth, in the frame of reference of the robot the observation covariance should rotate equally and in the opposite direction. This will induce rapid changes in the observation covariance—changes which ought to be predictable, but which will not be detected by an adaptive system.

2.3.3 State-dependent covariance models

Although the observation uncertainty may vary rapidly in time, it is reasonable to assume the uncertainty to vary slowly in space. This naturally induces a covariance model which is state-dependent.

$$\mathbf{R}_t = R(\mathbf{x}_t) \tag{2.43}$$

Stakkeland et al. [40] take this approach, formulating the noise parameters as explicit parametric functions of the state. This approach requires detailed domain knowledge in order to choose a reasonable model, as well as extensive tuning; for many domains it is not clear what form a reasonable noise model should take, or if one even exists. Aravkin and Burke [4] avoid the need for an explicit model through the use of arbitrary state-dependent noise functions for smoothing; by reformulating smoothing as an optimization problem, they show improvements in some limited contexts. However, the loss of the recursive structure makes this approach unsuitable for online use.

Ko and Fox [27] employ Gaussian processes in a filtering context to learn both the mean and covariance of the observation and process models directly from data. Like the adaptive models discussed in the previous section, they assume the observation and process distributions are stationary and Gaussian. The parameters of those distributions—the means and covariances—are permitted to vary with the state:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{x}}(\mathbf{x}_t), \boldsymbol{\Sigma}_{\mathbf{x}}(\mathbf{x}_t)\right) \tag{2.44}$$

$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{z}}(\mathbf{x}_t), \boldsymbol{\Sigma}_{\mathbf{z}}(\mathbf{x}_t)\right) \tag{2.45}$$

These mean and covariance functions are then learned using Gaussian processes. Provided the stationarity condition holds, in the limit of infinite samples the GP Bayes filter algorithm will obtain the true observation and process distributions. This method is of particular utility if the observation or process function is unknown or expressible only in terms of a large number of parameters; rather than choose a crude approximation, the non-parametric allows the function to be learned online.

The GP Bayes filter is fundamentally limited; it supports only diagonal covariances, rather than the more general positive definite cone. This limitation would seriously degrade performance in places where observation noise is expected to be correlated. For example, the uncertainty of observations from a scan matcher in a hallway is large in the direction parallel to the walls, regardless of which way the sensor is facing. The diagonal approximation is lacks the flexibility to describe this uncertainty, regardless of how slow the covariance changes or how much data is available.

An approach proposed by Melkumyan and Ramos [34] could mitigate this issue; they describe multiple-output Gaussian process kernels which generate symmetric positive definite covariance matrices. The approach is known as the multi-kernel Gaussian process. The work proposes a general method for constructing both the diagonal and cross-covariance terms from a restricted set of kernel functions. The set of valid covariance functions proposed includes sparse functions; this sparsity makes tractable the large matrix inversions necessary for Gaussian process inference.

Wilson and Ghahramani [43] present a non-parametric Bayesian method for estimating arbitrary distributions over positive definite matrices. The target application was volatility estimation, for use in quantitative finance; however, the method could be applied to sampling the covariances of the process and observation models, thus avoiding the need to learn the mean function while permitting the prediction of a full dense covariance matrix. However, because the method lacks a closed form solution, it is necessary to use sampling techniques to make predictions. This is prohibitively expensive for real-time use.

Although Ko and Fox make no provision for non-stationary distributions, adaptation to slowly-varying distribution parameters may be achieved by allowing the mean and covariance functions to depend on time.

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{x}}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\mathbf{x}}(\mathbf{x}_t, t)\right) \tag{2.46}$$

$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{z}}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\mathbf{z}}(\mathbf{x}_t, t)\right) \tag{2.47}$$

The Gaussian process would then favor more recent data over older data. The precise form of the Gaussian process kernel would determine the time scale over which the parameters could be permitted to vary.

Although this weakens the stationarity requirement, it does not remove the more fundamental limitation: the covariance of the observation or process models may not depend solely on the state, but also on the environment. This method lacks the flexibility to generalize to new environments; exploring new areas implies entering a region of new state space, where the GP filter has no information about noise characteristics.

2.3.4 Measurement-dependent covariance models

A time-dependent or state-dependent model cannot capture the dependence of observation uncertainty on the environment. However, the observations $\mathbf{z}_t$ are computed from sensor measurements $\boldsymbol{\zeta}_t$; it is natural to compute the observation covariance $\mathbf{R}$ based upon the same measurement.

$$\mathbf{R}_t = R(\boldsymbol{\zeta}_t) \tag{2.48}$$

Many authors have presented such measurement-dependent covariance models, often tailored to specific algorithms. Bengtsson and Baerveldt [10] approximate the covariance of observations derived from minimizing cost functions $C(\mathbf{z}, \boldsymbol{\zeta})$ which are well approximated as quadratic:

$$C(\mathbf{z}, \boldsymbol{\zeta}) = (\boldsymbol{\zeta} - \mathbf{M}\mathbf{z})^\top (\boldsymbol{\zeta} - \mathbf{M}\mathbf{z}) \tag{2.49}$$

Cost function such as this as are common in scan-matching odometry. The iterative closest point algorithm (ICP) computes a transform between two scans $\boldsymbol{\zeta}$ and $\boldsymbol{\zeta}'$, both represented in Cartesian coordinates, by choosing corresponding points in each scan and minimizes the squared error between corresponding points:

$$\mathbf{z} = \arg\min_{z} \|\boldsymbol{\zeta} - \mathbf{T}(z)\boldsymbol{\zeta}'\| \tag{2.50}$$

Here, $\mathbf{T}(z)$ is a rigid transform, incorporating a rotation and translation. Linearizing about some z yields a cost function of the form of equation (2.49), where $\mathbf{M}$ reflects the linearized transform. Under this approximation, the estimated covariance is

$$\operatorname{cov}(\mathbf{z}) \approx \left(\mathbf{M}^\top \mathbf{M}\right)^{-1} \sigma^2 = \left(\frac{1}{2}\frac{\partial^2 C}{\partial \mathbf{z}^2}\right)^{-1} \sigma^2 \tag{2.51}$$

with σ a measure of the measurement noise. This estimate is valid only when the cost function $C(\mathbf{z}, \boldsymbol{\zeta})$ has sufficiently small curvature for a linear approximation to be valid. As a corollary, equation (2.51) requires a smooth cost function; this precludes the use of the L^∞ norm, for example.

Andrea Censi [2] points out that this method and others like it do not consider the way the measurement affects the cost function C. He presents a method for estimating the covariance of any minimization algorithm, incorporating effects of both the measurement $\boldsymbol{\zeta}$ and the observation $\mathbf{z}$. In particular, he shows that for a cost function $C(\boldsymbol{\zeta}, \mathbf{z})$, the covariance can be approximated as

$$\operatorname{cov}(\mathbf{z}) \approx \left(\frac{\partial^2 C}{\partial \mathbf{z}^2}\right)^{-1} \frac{\partial^2 C}{\partial \mathbf{z} \partial \boldsymbol{\zeta}} \operatorname{cov}(\boldsymbol{\zeta}) \, \frac{\partial^2 C}{\partial \mathbf{z} \partial \boldsymbol{\zeta}}^\top \left(\frac{\partial^2 C}{\partial \mathbf{z}^2}\right)^{-1} \tag{2.52}$$

provided the map from measurements $\boldsymbol{\zeta}$ to observations $\mathbf{z}$ can be approximated by a first-order expansion. Applying this method to the iterative closest point algorithm for laser scan-matching odometry, he obtains a simple closed form estimate of the observation covariance.

Bachrach et al. [5] use a related cost function which first locates contours in the environment, then minimizes the distance to those contours. The shape of this error function induces a covariance matrix in much the same way as in the work of Bengtsson. However, the method of Bachrach implicitly incorporates measurement information through the contours, in contrast to the work of Censi, which incorporates that information explicitly.

Brenna [11] provides a useful survey, and a performance comparison of different approaches. All methods described here share an inherent weakness: they rely on the ability to obtain an accurate approximation of the cost function. If a linear or quadratic approximation is not accurate, the approximate predictive approach will generate inaccurate predictions. Moreover, there is little opportunity for learning or feedback; therefore if the predictions are initially suboptimal, they will remain suboptimal regardless of the amount of data collected.

Methods relying on the Jacobian or Hessian also require substantial programmer intervention to extend to new sensors—if such extension is even possible. For instance, one may desire to use an RGB-D camera instead of a laser range finder for odometry. In order to use the method of Censi, a programmer would need to derive the Hessian of the visual odometry system. It would be difficult to recycle code between sensors, despite the similarity of the problems. Any algorithmic tuning made for one sensor would likely need to be discarded.

A more general approach is to formulate the development of the covariance model as a learning problem. This is what Ko and Fox did for the state-dependent model, and their method could be extended to predict covariances based upon the raw measurement $\boldsymbol{\zeta}$.

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{x}}(\mathbf{x}_t, \boldsymbol{\zeta}_t), \boldsymbol{\Sigma}_{\mathbf{x}}(\mathbf{x}_t, \boldsymbol{\zeta}_t)\right) \tag{2.53}$$

$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{z}}(\mathbf{x}_t, \boldsymbol{\zeta}_t), \boldsymbol{\Sigma}_{\mathbf{z}}(\mathbf{x}_t, \boldsymbol{\zeta}_t)\right) \tag{2.54}$$

Because the mean function is known for many domains, I restrict the learning to just the noise parameters; moreover, for simplicity I focus on learning the observation covariance, noting that the same general procedure could be used to learn the process covariance. Under this model, the covariance will be predicted, rather than adapted; this allows for rapid changes, while still permitting generalization to new environments. In the next chapter, I explore a method for learning models of this form.

Chapter 3

Covariance Prediction

Recall the motivating example of a robot with state $\mathbf{x}_t \in \mathbb{R}^{N_\mathbf{x}}$ at time t, with $N_\mathbf{x}$ the dimensionality of the state vector. It is equipped with a sensor providing noisy raw measurements $\boldsymbol{\zeta}_t \in \mathbb{R}^{N_\zeta}$ at each time step; these measurements could be the pixel values of a camera, for instance, or the ranges returned by a planar LIDAR unit, and as such the measurement dimensionality $N_{\boldsymbol{\zeta}}$ may be quite large. The raw sensor measurements are dependent on both the (unobserved) robot state $\mathbf{x}_t$ and on the environment Ω. Given this probabilistic model of the sensor data, shown in figure 2-1a, as well as a model of the state dynamics, the history of measurements can be used to infer the state of the robot.

To avoid reasoning about the environment, we compute an observation $\mathbf{z}_t = \mathbf{z}(\boldsymbol{\zeta}_t)$ from the high-dimensional signal available from sensors. We model the observation as a multivariate Gaussian, with a mean defined by a known deterministic *observation function*, $h\left(\mathbf{x}\right)$, and a *covariance function* $R(\boldsymbol{\zeta}_t)$. This covariance function $R : \boldsymbol{\zeta} \mapsto \mathbf{R}$ maps a measurement $\boldsymbol{\zeta} \in \mathbb{R}^{N_\zeta}$ to a symmetric positive definite matrix $\mathbf{R} \in \mathbb{R}^{N_\mathbf{z} \times N_\mathbf{z}} \succ \mathbf{0}$. Our observation distribution is then given by equation (3.1).

$$\mathbf{z}_t \sim \mathcal{N}\left(h\left(\mathbf{x}_t\right), R(\boldsymbol{\zeta}_t)\right). \tag{3.1}$$

Our goal is to learn $R(\boldsymbol{\zeta})$ given a set of sample data, in order to predict a covariance $\hat{\mathbf{R}}$ for a given observation. To facilitate the learning process, we assume that the covariance function depends on the measurement $\boldsymbol{\zeta}$ only through a set of deterministic predictor features $\boldsymbol{\phi}$. This assumption results in no loss of generality; if we choose the predictor features to be the elements of the measurement vector $\boldsymbol{\zeta}$ we exactly recover the formulation of equation (3.1). However, because there is often a set of features such that $N_{\boldsymbol{\phi}} \ll N_{\boldsymbol{\zeta}}$, we may learn a function from $\mathbb{R}^{N_{\boldsymbol{\phi}}} \to \mathbb{R}^{N_{\mathbf{z}} \times N_{\mathbf{z}}} \succ \mathbf{0}$, which is a far easier learning problem. This implies the observation distribution given in equation (3.2).

$$\mathbf{z}_t \sim \mathcal{N}\left(h\left(\mathbf{x}_t\right), R(\boldsymbol{\phi}_t)\right). \tag{3.2}$$

We assume, for the moment, that we are given both the measurements $\boldsymbol{\zeta}$ and the corresponding states $\mathbf{x}$. From these measurements, we compute observations $\mathbf{z}$ and predictors $\boldsymbol{\phi}$. Our problem, formally stated, is then to learn an estimate $\hat{\mathbf{R}}(\boldsymbol{\phi})$ of the covariance function $R(\boldsymbol{\phi})$ given a set $\mathcal{D}$ of N_D triplets $(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\phi}_i) \quad \forall i \in [1, N_D]$.

3.1 Estimating a fixed covariance

Recall that if $\mathbf{v} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}\right)$, then

$$\mathbf{E}\left[\mathbf{v}\mathbf{v}^\top\right] = \mathbf{R} \tag{3.3}$$

If we have a set of N independent samples $\{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ from a multivariate Gaussian, so that $\mathbf{v}_i \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}\right) \quad \forall i \in [1, N]$, then both the maximum likelihood estimate and the minimum variance unbiased estimate of the distribution covariance is the empirical covariance.

$$\hat{\mathbf{R}} = \frac{1}{N} \sum_{i}^{N} \mathbf{v}_i \mathbf{v}_i^\top \tag{3.4}$$

For a linear Gaussian measurement model, $h\left(\mathbf{x}\right) = H\mathbf{x} + \mathbf{v}$ with $\mathbf{v} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}\right)$, we may compute samples $\mathbf{v} = \mathbf{z} - H\mathbf{x}$ if we have access to a sequence of both measurements $\mathbf{z}$ and states $\mathbf{x}$. Therefore, if we have $\{(\mathbf{x}_1, \mathbf{z}_1), \ldots, (\mathbf{x}_N, \mathbf{z}_N)\}$, the minimum variance estimate $\hat{\mathbf{R}}$ for the observation covariance $\mathbf{R}$ is the outer product.

$$\hat{\mathbf{R}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{z}_i - H\mathbf{x}_i)(\mathbf{z}_i - H\mathbf{x}_i)^\top = \frac{1}{N}\sum_{i=1}^{N}\mathbf{T}_{\mathbf{R}}(\mathbf{x}_i, \mathbf{z}_i) \tag{3.5}$$

We introduce the notation $\mathbf{T}_{\mathbf{R}}(\mathbf{x}_i, \mathbf{z}_i) = (\mathbf{z}_i - H\mathbf{x}_i)(\mathbf{z}_i - H\mathbf{x}_i)^\top$ for brevity in future derivations. Importantly, if the covariance is fixed, this estimator is unbiased.

$$\mathbf{E}\left[\hat{\mathbf{R}}\right] = \mathbf{E}\left[\mathbf{T}_{\mathbf{R}}(\mathbf{x}, \mathbf{z})\right] = \mathbf{R} \tag{3.6}$$

3.2 Kernel Estimation

If the covariance $\mathbf{R}$ is free to vary with the predictor, equation (3.6) is true only when conditioned on the predictor vector $\boldsymbol{\phi}$.

$$R(\boldsymbol{\phi}_t) = \mathbf{E}\left[\mathbf{T}_{\mathbf{R}}|\boldsymbol{\phi}_t\right] = \int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, \mathbf{T}_{\mathbf{R}}(\mathbf{x}, \mathbf{z}) p(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi}_t) \tag{3.7}$$

We may rewrite equation (3.7) in terms of the joint distribution $p(\mathbf{x}, \mathbf{z}, \boldsymbol{\phi})$:

$$\begin{aligned}
\mathbf{E}\left[\mathbf{T}_{\mathbf{R}}|\boldsymbol{\phi}\right] &= \frac{\int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, \mathbf{T}_{\mathbf{R}}(\mathbf{x}, \mathbf{z}) p(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi})\, p(\boldsymbol{\phi})}{\int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi})\, p(\boldsymbol{\phi})} && (3.8) \\
&= \frac{\int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, \mathbf{T}_{\mathbf{R}}(\mathbf{x}, \mathbf{z}) p(\mathbf{x}, \mathbf{z}, \boldsymbol{\phi})}{\int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}, \mathbf{z}, \boldsymbol{\phi})} && (3.9)
\end{aligned}$$

This is beneficial, because given a set $\mathcal{D}$ of observations,

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\phi}_i) \quad \forall i \in [1, N_D]\} \tag{3.10}$$

we may approximate that joint distribution using kernel regression techniques.

A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is an admissible *kernel* if it has the following properties.

$$\begin{array}{rrr}
\textbf{Non-negative:} & k(\mathbf{x},\mathbf{x}') \geq 0 & \forall \mathbf{x},\mathbf{x}' \\
\textbf{Normalized:} & \int_{\mathcal{X}} \mathrm{d}\mathbf{x}' k(\mathbf{x},\mathbf{x}') = 1 & \forall \mathbf{x} \\
\textbf{Asymptotically identical:} & \lim_{N_D \to \infty} k(\mathbf{x},\mathbf{x}') = \delta(\mathbf{x}-\mathbf{x}') & \forall \mathbf{x},\mathbf{x}' \\
\textbf{Exponentially bounded:} & \lim_{\|\mathbf{x}-\mathbf{x}'\| \to \infty} \exp(\lambda \|\mathbf{x}-\mathbf{x}'\|) k(\mathbf{x},\mathbf{x}') < \infty & \exists \lambda > 0, \forall \mathbf{x},\mathbf{x}'
\end{array}$$

It can be shown that the kernel density estimate

$$\hat{p}(\mathbf{x}) = \frac{1}{N_D} \sum_{i=1}^{N_D} k(\mathbf{x},\mathbf{x}_i). \tag{3.11}$$

is a consistent estimate of the distribution $p(\mathbf{x})$; see appendix B for a proof. It follows that we may approximate the distribution $p(\mathbf{x},\mathbf{z},\boldsymbol{\phi})$ by a sum of kernel functions.

$$p(\mathbf{x},\mathbf{z},\boldsymbol{\phi}) \approx \hat{p}\left(\mathbf{x},\mathbf{z},\boldsymbol{\phi}\right) = \frac{1}{N} \sum_{i=1}^{N} k(\mathbf{x},\mathbf{x}_i) k(\mathbf{z},\mathbf{z}_i) k(\boldsymbol{\phi},\boldsymbol{\phi}_i) \tag{3.12}$$

I restrict attention to isotropic, symmetric kernels:

$$k(\mathbf{x}_i,\mathbf{x}_j) = k(\|\mathbf{x}_i - \mathbf{x}_j\|, \sigma_{\mathbf{x}}) \tag{3.13}$$

and likewise for the observation and predictor kernels. The kernel scale σ is a scalar defining the size of the kernel, defined such that $k(\|\mathbf{x}\|,\sigma) = \sigma k(\sigma\|\mathbf{x}\|, 1)$. The norm $\|\mathbf{x}_i - \mathbf{x}_j\|$ will often be a scaled Euclidean norm:

$$\|\mathbf{x}\|_{\mathbf{M}} = \sqrt{\mathbf{x}^\top \mathbf{M} \mathbf{x}} \tag{3.14}$$

This form is not essential, but simplifies analysis, and experimentally the use of other norms has little effect on performance. Note that this choice is minimally restrictive; provided the space of predictors can be embedded in the space of real numbers, the analysis presented here is valid.

Common kernels include the linear kernel,

$$k(\|\Delta\mathbf{x}\|, 1) \propto (1 - \|\Delta\mathbf{x}\|)\mathbb{1}_{\|\Delta\mathbf{x}\|<1}, \tag{3.15}$$

the quadratic kernel,

$$k(\|\Delta\mathbf{x}\|, 1) \propto (1 - \|\Delta\mathbf{x}\|^2)\mathbb{1}_{\|\Delta\mathbf{x}\|<1}, \tag{3.16}$$

and the squared exponential, or Gaussian, kernel,

$$k(\|\Delta\mathbf{x}\|, 1) \propto \exp(-\|\Delta\mathbf{x}\|^2). \tag{3.17}$$

Kernels with finite support, such as the linear or quadratic kernels, offer an important computational advantage: the distribution near a point $\mathbf{x}$ will only be influenced by a subset of the available samples. There exist data structures designed for efficiently finding all points within a fixed distance of a specified query point; for instance, the k-D tree can find nearby points in time logarithmic in the number of available samples. For large datasets, this speedup can enormous.

Using our isotropic kernel, we may approximate the joint distribution of equation (3.9) as

$$\hat{p}\left(\mathbf{x}, \mathbf{z}, \boldsymbol{\phi}\right) = \frac{1}{N}\sum_{i=1}^{N} k(\|\mathbf{x} - \mathbf{x}_i\|, \sigma_{\mathbf{x}}) k(\|\mathbf{z} - \mathbf{z}_i\|, \sigma_{\mathbf{z}}) k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|, \sigma_{\boldsymbol{\phi}}) \tag{3.18}$$

This approximation allows us to evaluate the integral in equation (3.9). Substituting the approximation of equation (3.18) for the true distribution, we find

$$\begin{aligned} \mathbf{E}\left[\mathbf{T}_{\mathbf{R}}|\boldsymbol{\phi}\right] &= \frac{\int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, \mathbf{T}_{\mathbf{R}}(\mathbf{x}, \mathbf{z}) p(\mathbf{x}, \mathbf{z}, \boldsymbol{\phi})}{\int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}, \mathbf{z}, \boldsymbol{\phi})} \\ &= \frac{\int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, \mathbf{T}_{\mathbf{R}}(\mathbf{x}, \mathbf{z}) \frac{1}{N}\sum_{i=1}^{N} k(\|\mathbf{x} - \mathbf{x}_i\|, \sigma_{\mathbf{x}}) k(\|\mathbf{z} - \mathbf{z}_i\|, \sigma_{\mathbf{z}}) k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|, \sigma_{\boldsymbol{\phi}})}{\int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, \frac{1}{N}\sum_{i=1}^{N} k(\|\mathbf{x} - \mathbf{x}_i\|, \sigma_{\mathbf{x}}) k(\|\mathbf{z} - \mathbf{z}_i\|, \sigma_{\mathbf{z}}) k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|, \sigma_{\boldsymbol{\phi}})} \end{aligned} \tag{3.19}$$

The constant factors of $\frac{1}{N}$ cancel; properties of the integral allow us to reverse the order of summation and integration.

$$\mathbf{E}\left[\mathbf{T_R}|\boldsymbol{\phi}\right] = \frac{\sum_{i=1}^{N} \int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, \mathbf{T_R}(\mathbf{x},\mathbf{z}) k(\|\mathbf{x}-\mathbf{x}_i\|,\sigma_{\mathbf{x}}) k(\|\mathbf{z}-\mathbf{z}_i\|,\sigma_{\mathbf{z}}) k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma_{\boldsymbol{\phi}})}{\sum_{i=1}^{N} \int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, k(\|\mathbf{x}-\mathbf{x}_i\|,\sigma_{\mathbf{x}}) k(\|\mathbf{z}-\mathbf{z}_i\|,\sigma_{\mathbf{z}}) k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma_{\boldsymbol{\phi}})} \tag{3.20}$$

Because the kernel functions are required to be normalized, we may immediately simplify the expression in the denominator.

$$\mathbf{E}\left[\mathbf{T_R}|\boldsymbol{\phi}\right] = \frac{\sum_{i=1}^{N} \int \mathrm{d}\mathbf{z} \int \mathrm{d}\mathbf{x}\, \mathbf{T_R}(\mathbf{x},\mathbf{z}) k(\|\mathbf{x}-\mathbf{x}_i\|,\sigma_{\mathbf{x}}) k(\|\mathbf{z}-\mathbf{z}_i\|,\sigma_{\mathbf{z}}) k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma_{\boldsymbol{\phi}})}{\sum_{i=1}^{N} k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma_{\boldsymbol{\phi}})} \tag{3.21}$$

Finally, note that we are free to choose the state kernel function $k(\|\mathbf{x}-\mathbf{x}_i\|,\sigma_{\mathbf{x}})$ and the observation kernel function $k(\|\mathbf{z}-\mathbf{z}_i\|,\sigma_{\mathbf{z}})$ as we see fit. In order to recover the fixed covariance estimator in the special case when the covariance is independent from the predictor vector, the unique choice for the state and observation kernel functions is the Dirac delta function, $k(\|\mathbf{x}-\mathbf{x}_i\|,\sigma_{\mathbf{x}}) = \delta(\|\mathbf{x}-\mathbf{x}_i\|)$, and $k(\|\mathbf{z}-\mathbf{z}_i\|,\sigma_{\mathbf{z}}) = \delta(\|\mathbf{z}-\mathbf{z}_i\|)$ Evaluating the integrals for this choice of kernels, we arrive at a closed-form estimator.

$$\mathbf{E}\left[\mathbf{T_R}|\boldsymbol{\phi}\right] = \frac{\sum_{i=1}^{N} \mathbf{T_R}(\mathbf{x}_i,\mathbf{z}_i) k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma_{\boldsymbol{\phi}})}{\sum_{i=1}^{N} k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma_{\boldsymbol{\phi}})} \tag{3.22}$$

This is a Nadaraya-Watson estimator [36, 42], extended to estimate the unobservable quantity $\mathbf{R}$ in terms of the observable statistic $\mathbf{E}\left[\mathbf{T_R}\right]$. The kernel function allows us to compute an expected covariance by averaging over a set of nearby, but not identical, measurements in the data set. This is crucial; it is almost certain[1] we will not measure the exact same predictor $\boldsymbol{\phi}$ twice, since our predictors are continuous functions.

Note that the final estimate depends only on the predictor $\boldsymbol{\phi}$; we have eliminated the dependence on the state $\mathbf{x}$ and the observation $\mathbf{z}$. This independence from the state is encoded in the model structure that underlies the algorithm, and its emergence here was inevitable. Importantly, the estimate is now independent of the

[1] In the probabilistic sense: the set of predictors with $\boldsymbol{\phi} = \boldsymbol{\phi}_0$ has measure zero.

environment given the predictor vector, and thus can trivially generalize to new environments provided the predictor encodes the relevant information contained in the raw measurement.

This defines a procedure, described in detail in algorithm 1, for predicting the covariance of a new observation at run time, given a dataset $\mathcal{D}$ and a prescribed kernel function. First, compute the observation $\mathbf{z}$ and predictor vector $\boldsymbol{\phi}$ for the new raw measurement. Then, evaluate the kernel function $k(\boldsymbol{\phi}, \boldsymbol{\phi}_i)$ for each sample in the available data set. The prediction is then given by equation (3.22) as a sum of outer products of error vectors weighted by the kernel function.

Algorithm 1 Covariance prediction

```
Input: Query point $\boldsymbol{\phi}$
Input: Dataset $\mathcal{D} = \{(\mathbf{v}_i, \boldsymbol{\phi}_i) \quad \forall i \in [1, N_D]\}$
Input: Parameters $\boldsymbol{\alpha}$ of kernel function $k(\cdot,\cdot)$
Output: Estimate $\hat{\mathbf{R}}$ of the covariance $\mathbf{R}$ at point $\boldsymbol{\phi}$
  function PredictCovariance($\boldsymbol{\phi}, \mathcal{D}, \boldsymbol{\alpha}$)
      $\hat{\mathbf{R}} \leftarrow \mathbf{0}$, $n \leftarrow 0$
      $\mathcal{S} \leftarrow$Neighbors($\boldsymbol{\phi}$)          ▷ Returns $\mathcal{S} \subset \mathcal{D} : \|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|_{\mathbf{M}} < \sigma \quad \forall i \in \mathcal{S}$
      for $i \in \mathcal{S}$ do
          $\hat{\mathbf{R}} \leftarrow \hat{\mathbf{R}} + k(\boldsymbol{\phi}, \boldsymbol{\phi}_i)\mathbf{v}_i\mathbf{v}_i^\top$
          $n \leftarrow n + k(\boldsymbol{\phi}, \boldsymbol{\phi}_i)$
      end for
      $\hat{\mathbf{R}} \leftarrow \frac{1}{n}\hat{\mathbf{R}}$
      return $\hat{\mathbf{R}}$
  end function
```

3.3 Asymptotic Properties

I first present several asymptotic properties of the kernel estimator, and demonstrate that there exist kernel parameters for which performance is guaranteed. A derivation of these properties is available in appendix B. Let the vector $\boldsymbol{\theta}$ represent the independent elements of the matrix $\mathbf{R}$, and the elements of the vector estimator $\mathbf{T}_{\boldsymbol{\theta}}$ represent the corresponding elements of the above matrix estimator $\mathbf{T}_{\mathbf{R}}$. Using this notation, the estimator in equation (3.22) can be written

$$\hat{\boldsymbol{\theta}}(\boldsymbol{\phi}) = \frac{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|)\mathbf{T}_{\boldsymbol{\theta}}(\mathbf{v}_i)}{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|)} \tag{3.23}$$

I denote the kernel scale as σ and assume the kernel metric to be of *generalized Euclidean form*, with a metric tensor $\mathbf{M}$:

$$\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\| = \sqrt{(\boldsymbol{\phi} - \boldsymbol{\phi}_i)^\top \mathbf{M} (\boldsymbol{\phi} - \boldsymbol{\phi}_i)} \tag{3.24}$$

This permits the definition of a local coordinate system $\boldsymbol{\varphi} = \frac{1}{\sigma}\mathbf{L}(\boldsymbol{\phi} - \boldsymbol{\phi}_i)$, where $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$; in this local coordinate system, the kernel function is spherically symmetric.

$$k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|_{\mathbf{M}}, \sigma) = \sigma k(\|\boldsymbol{\varphi}\|) \tag{3.25}$$

With these assumptions, it can be shown that, in the limit of many samples and small scale, the estimator is unbiased to first order.

$$\lim_{\substack{\sigma \to 0 \\ N\sigma \to \infty}} \mathbf{E}\left[\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}(\boldsymbol{\phi})\right] = \left(\nabla^2_{\mathbf{M}} \boldsymbol{\theta}(\boldsymbol{\phi}) + 2\nabla \log p(\boldsymbol{\phi})^\top \mathbf{M}^{-1} \nabla \boldsymbol{\theta}(\boldsymbol{\phi}) + \nabla^2_{\mathbf{M}} p(\boldsymbol{\phi})\, \boldsymbol{\theta}(\boldsymbol{\phi})\right) \frac{\sigma^2 c_K}{2} \tag{3.26}$$

Here, $\nabla^2_{\mathbf{M}} \boldsymbol{\theta}_i(\boldsymbol{\phi}) = \operatorname{tr}\left(\nabla \nabla^\top \boldsymbol{\theta}_i(\boldsymbol{\phi}) \mathbf{M}^{-1}\right)$ is the Laplacian under the metric $\mathbf{M}$, and $c_k = \int \varphi_i^2 k(\boldsymbol{\varphi}) \mathrm{d}\boldsymbol{\varphi}$ is the second moment of any element under the kernel, since each element is treated identically. In the special case of the Euclidean metric, where $\mathbf{M} = \mathbb{1}$, and uniform sampling density, this can be reduced further.

$$\lim_{\substack{\sigma \to 0 \\ N\sigma \to \infty}} \mathbf{E}\left[\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}(\boldsymbol{\phi})\right] = \frac{1}{2} \nabla^2 \boldsymbol{\theta}_i(\boldsymbol{\phi}) \sigma^2 c_K \tag{3.27}$$

In addition, the same assumptions give an asymptotic variance.

$$\lim_{\substack{\sigma \to 0 \\ N\sigma \to \infty}} \mathbf{V}\left(\hat{\boldsymbol{\theta}}\right) = \frac{d_K}{p(\boldsymbol{\phi})\, N_D \sigma} \mathbf{V}\left(\boldsymbol{\theta}_i | \boldsymbol{\phi}\right) \tag{3.28}$$

Here, $d_k = \int k(\boldsymbol{\varphi})^2 \mathrm{d}\boldsymbol{\varphi}$. If we define the kernel scale σ as a function of N_D, we may express equations (3.26) and (3.28) in terms of just a single condition. Suppose $\sigma \propto N_D^{-\alpha}$. The bias and variance may then be expressed as

$$\lim_{N_D \to \infty} \mathbf{E}\left[\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}(\boldsymbol{\phi})\right] \propto \left(\frac{1}{2}\nabla^2_{\mathbf{M}}\boldsymbol{\theta}_i(\boldsymbol{\phi}) + \nabla \log p(\boldsymbol{\phi})^\top \mathbf{M}^{-1}\nabla\boldsymbol{\theta}(\boldsymbol{\phi})\right) N_D^{-2\alpha} c_K \tag{3.29}$$

and

$$\lim_{N_D \to \infty} \mathbf{V}\left(\hat{\boldsymbol{\theta}}\right) \propto \frac{d_K}{p(\boldsymbol{\phi})} N_D^{\alpha-1} \mathbf{V}\left(\boldsymbol{\theta}_i | \boldsymbol{\phi}\right) \tag{3.30}$$

Provided $0 < \alpha < 1$, in the limit as N_D goes to infinity both the bias and variance become zero; we have obtained a consistent estimator for the covariance $\mathbf{R}$.

3.4 CELLO

In practice, data sets will be finite. For a fixed amount of available data, the choice of kernel and metric scale parameters will influence both the bias and the variance of the estimator. Under a scaled Euclidean metric and an isometric kernel, these parameters are a metric tensor $\mathbf{M} \in \mathbb{R}^{N_\phi \times N_\phi} \succ \mathbf{0}$ and a kernel scale $\sigma \in \mathbb{R}$.[2] The metric tensor $\mathbf{M}$ determines the *relative* importance of the elements of the predictor vector $\boldsymbol{\phi}$ in determining the distance between samples, and the kernel scale σ determines how close together samples must be for their expected error to be strongly correlated. If these parameters of the kernel function dictate that two sample predictors $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$ are far apart, the predicted covariance at the query point $\boldsymbol{\phi}$ will not depend strongly on the observed error at $\boldsymbol{\phi}'$. On the other hand, if the parameters determine that $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$ are very close, then the measured error at $\boldsymbol{\phi}'$ will contribute strongly to the predicted error at $\boldsymbol{\phi}$.

By choosing $\mathbf{M}$ to make $\nabla^2_{\mathbf{M}}\boldsymbol{\theta}_i(\boldsymbol{\phi})$ as small as possible, we may mitigate estimator bias; this permits a smaller σ and hence a smaller estimator variance. Without

[2] Neither the prediction process nor the learning process require a scaled Euclidean metric or an isometric single-parameter kernel; future work includes investigating the potential of non-Euclidean metrics. Assuming a Euclidean metric parameterized by a symmetric tensor facilitates understanding what the learning process is accomplishing.

knowing the curvature of the expected parameter manifold—as is always the case, in practice—we cannot choose $\{\sigma, \mathbf{M}\}$ analytically. Even minimizing equations (3.26) and (3.28) numerically would necessarily be a complex, iterative process, involving third derivatives of non-parametric terms.

We avoid this problem by noting that minimizing bias and variance is not our the ultimate goal; our objective is to identify a sensor observation model. The metric and scale can be treated as the parameters of a model; the best model parameters are the parameters that maximize the likelihood of the observations $\mathbf{z}_{1:N_D}$, given the corresponding state vectors $\mathbf{x}_{1:N_D}$. To simplify notation, we represent the elements of the metric and scale parameters by a single vector of *hyperparameters* $\boldsymbol{\alpha} \in \mathbb{R}^{N_{\boldsymbol{\alpha}}}$.

Equation (3.22) provides a simple form for making predictions given a hyperparameter vector $\boldsymbol{\alpha}$.

$$\hat{\mathbf{R}}(\boldsymbol{\phi}, \boldsymbol{\alpha}) = \frac{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|, \boldsymbol{\alpha})\mathbf{v}_i\mathbf{v}_i^\top}{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|, \boldsymbol{\alpha})} \tag{3.31}$$

where as before, $\mathbf{v}_i = \mathbf{z}_i - \mathbf{H}\mathbf{x}_i$ is a vector of observation noise. The likelihood of the parameters of a Gaussian observation is equal to the probability of the observation given the parameters.

$$l(\mathbf{R}_i|\mathbf{v}_i) = \mathcal{N}(\mathbf{v}_i|\mathbf{0}, \mathbf{R}_i) = \frac{1}{\sqrt{(2\pi)^{N_\mathbf{z}} \det \mathbf{R}_i}} \exp(-\frac{1}{2}\mathbf{v}_i^\top \mathbf{R}_i^{-1} \mathbf{v}_i) \tag{3.32}$$

Given N_D independent observations and an observation covariance parameterized by the hyperparameters as in equation (3.31), the likelihood of a hyperparameter vector $\boldsymbol{\alpha}$ may be expressed analogously.

$$l(\boldsymbol{\alpha}|\mathcal{D}) = \prod_{i=1}^{N_D} \frac{1}{\sqrt{(2\pi)^{N_\mathbf{z}} \det \hat{\mathbf{R}}(\boldsymbol{\phi}_i, \boldsymbol{\alpha})}} \exp(-\frac{1}{2}\mathbf{v}_i^\top \hat{\mathbf{R}}(\boldsymbol{\phi}_i, \boldsymbol{\alpha})^{-1} \mathbf{v}_i) \tag{3.33}$$

It is more convenient to work with the logarithm of this likelihood; the monotonicity of the logarithm implies the maximum likelihood hyperparameters and the maximum log likelihood hyperparameters are identical. Ignoring constant factors, the log likelihood

$\mathcal{L}$ may be written as a simple sum.

$$\mathcal{L}(\boldsymbol{\alpha}|\mathcal{D}) = -\frac{1}{2}\sum_{i=1}^{N_D}\left(\log\det\hat{\mathbf{R}}(\boldsymbol{\phi}_i,\boldsymbol{\alpha}) + \mathbf{v}_i^\top\hat{\mathbf{R}}(\boldsymbol{\phi}_i,\boldsymbol{\alpha})^{-1}\mathbf{v}_i\right) \tag{3.34}$$

The best observation model is the one that maximizes the likelihood of the observations; we therefore choose the hyperparameters that maximize this likelihood function.

$$\boldsymbol{\alpha}^\star = \arg\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) \tag{3.35}$$

This is a straightforward optimization problem, and may be solved numerically using well-known techniques.

Note the objective function has a closed form Jacobian, which may be use to accelerate the optimization:

$$\begin{aligned}\nabla\mathcal{L}(\boldsymbol{\alpha}) &= -\frac{1}{2}\sum_{i=1}^{N_D}\operatorname{tr}\left(\hat{\mathbf{R}}^{-1}\nabla\hat{\mathbf{R}}\right) - \mathbf{v}_i^\top\hat{\mathbf{R}}^{-1}\nabla\hat{\mathbf{R}}\hat{\mathbf{R}}^{-1}\mathbf{v}_i && (3.36)\\ &= -\frac{1}{2}\sum_{i=1}^{N_D}\sum_{j=1}^{N_D}\nabla k_{ij}\mathbf{v}_j^\top\hat{\mathbf{R}}(\boldsymbol{\phi}_i,\boldsymbol{\alpha})^{-1}(\hat{\mathbf{R}}(\boldsymbol{\phi}_i,\boldsymbol{\alpha}) - \mathbf{v}_i\mathbf{v}_i^\top)\hat{\mathbf{R}}(\boldsymbol{\phi}_i,\boldsymbol{\alpha})^{-1}\mathbf{v}_j && (3.37)\end{aligned}$$

where

$$\nabla k_{ij} = \nabla\frac{k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\boldsymbol{\alpha})}{\sum_{i=1}^{N_D}k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\boldsymbol{\alpha})} \tag{3.38}$$

Examining the Jacobian reveals an interesting property. First, there is one closed-form solution:

$$\hat{\mathbf{R}}(\boldsymbol{\phi}_i,\boldsymbol{\alpha}) = \mathbf{v}_i\mathbf{v}_i^\top \quad \forall i \tag{3.39}$$

This is achieved for $\sigma = 0$, as

$$\lim_{\sigma\to 0} k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma) = \delta(\boldsymbol{\phi}-\boldsymbol{\phi}_i) \tag{3.40}$$

for any of the isotropic decreasing kernels described. However, this solution will generate a valid prediction only for predictors that have been previously observed, and even for those samples will yield a high-variance estimate. Avoiding this zero solution ensures better generalization, and may be done by employing ‘leave one out’ validation; that is, by computing the covariance estimate $\hat{\mathbf{R}}(\boldsymbol{\phi}_i)$, using all samples but sample i:

$$\hat{\mathbf{R}}(\boldsymbol{\phi}_i, \boldsymbol{\alpha}) = \frac{\sum_{j \neq i} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_j\|, \boldsymbol{\alpha}) \mathbf{v}\mathbf{v}^\top}{\sum_{j \neq i} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_j\|, \boldsymbol{\alpha})} \tag{3.41}$$

Our algorithm for learning the kernel parameters to maximize the modified objective of equation (3.41) is called Covariance Estimation through Learned Likelihood Optimization, or CELLO, and is summarized in algorithm 2. This algorithm makes use of the covariance prediction subroutine of algorithm 1; the kernel function used in that subroutine should be evaluated using the current hyperparameter estimate $\boldsymbol{\alpha}$. As presented, the algorithm employs naïve gradient descent for optimization, using random restarts to avoid suboptimal local maxima. The number of restarts and the learning rate $\boldsymbol{\eta}$ both must be tuned for performance; nominally, the learning rate $\boldsymbol{\eta}$ should be as small as possible, and the number of restarts as large as feasible. In practice it is often helpful to make the learning rate initially very small and slowly increase it until optimizer performance degrades. Although the gradient descent variant is simple to explain and implement, more complex optimization techniques may also be employed; the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method outperformed simple gradient descent in many of the applications considered.

3.5 Learning without ground truth

The learning process described, CELLO, assumes that for each predictor $\boldsymbol{\phi}_i$ in our data set we have a corresponding observation noise $\mathbf{v}_i = \mathbf{z}_i - h(\mathbf{x}_i)$. By construction, the expected observation is a function of the vehicle state; if we know the state at the time an observation was taken, we may compute the observation noise, defined as the difference between the actual observation and its expectation. The true vehicle state can be acquired using a variety of techniques, including external measurement systems such as motion capture; often, however, these approaches are difficult to implement. If the state is unavailable, the noise vector $\mathbf{v}_i$ cannot be computed and the machinery of CELLO cannot be applied. In this section, we extend CELLO to function without

Algorithm 2 Covariance estimation through log likelihood optimization (CELLO)

```
Input: Dataset $\mathcal{D} = \{(\mathbf{v}_i, \boldsymbol{\phi}_i) \quad \forall i \in [1, N_D]\}$
Output: Kernel parameters $\boldsymbol{\alpha}^\star = \arg\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}|\mathcal{D})$
  function MaximizeLikelihood($\mathcal{D}$)
      Set learning rate $\boldsymbol{\eta}$
      Set number of restarts $m$
      for $i = [1, m]$ do
          Randomly initialize $\boldsymbol{\alpha}$
          repeat
              for $j \in [1, N_D]$ do
                  $\hat{\mathbf{R}}_j \leftarrow$PredictCovariance($\boldsymbol{\phi}_j, \mathcal{D}, \boldsymbol{\alpha}$)
              end for
              $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \boldsymbol{\eta} \nabla \mathcal{L}\left(\hat{\mathbf{R}}_{1:N_D}, \mathcal{D}\right)$
          until convergence
          if $\mathcal{L}(\boldsymbol{\alpha}|\mathcal{D}) > \mathcal{L}(\boldsymbol{\alpha}^\star|\mathcal{D})$ then
              $\boldsymbol{\alpha}^\star \leftarrow \boldsymbol{\alpha}$
          end if
      end for
      return $\boldsymbol{\alpha}^\star$
  end function
```

knowledge of state, by using sensor observations and a process model to infer the unobserved state.

Recall the joint distribution over states and observations may be written

$$p(\mathbf{x}_{1:N_D}, \mathbf{z}_{1:N_D}) = p(\mathbf{x}_0) \prod_{i=1}^{N_D} p(\mathbf{z}_i|\mathbf{x}_i)\, p(\mathbf{x}_i|\mathbf{x}_{i-1}) \tag{3.42}$$

$$= \mathcal{N}\left(\mathbf{x}_0|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\right) \prod_{i=1}^{N_D} \mathcal{N}\left(\mathbf{z}_i|h\left(\mathbf{x}_i\right), \mathbf{R}_i\right) \mathcal{N}\left(\mathbf{x}_i|f\left(\mathbf{x}_{i-1}\right), \mathbf{Q}_i\right) \tag{3.43}$$

If we know the covariances $\mathbf{R}_{1:N_D}$ and $\mathbf{Q}_{1:N_D}$, we can infer a distribution over each latent state using the Kalman filter framework, as described in chapter 2.

$$p(\mathbf{x}_i|\mathbf{z}_{1:N_D}) = \mathcal{N}\left(\mathbf{x}_i|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \quad \forall i \in [1, N_D] \tag{3.44}$$

Recall from equation (3.6) that in the case of known vehicle states and fixed observation covariance, the outer product of the observation noise $\mathbf{v}_i = \mathbf{z}_i - h\left(\mathbf{x}_i\right)$ was found to be an unbiased estimator of the observation covariance. Taking the expectation of the same estimate for a normally distributed vehicle state yields a biased estimate.

$$\begin{aligned}\mathbf{E}\left[(\mathbf{z}-\mathbf{H}\mathbf{x})(\mathbf{z}-\mathbf{H}\mathbf{x})^\top\right] &= \int \mathrm{d}\mathbf{x} \int \mathrm{d}\mathbf{z}\, (\mathbf{z}-\mathbf{H}\mathbf{x})(\mathbf{z}-\mathbf{H}\mathbf{x})^\top \mathcal{N}\left(\mathbf{z}|\mathbf{H}\mathbf{x},\mathbf{R}\right) \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right) \\ &= \mathbf{R} + \mathbf{H}\boldsymbol{\Sigma}\mathbf{H}^\top \end{aligned} \tag{3.45}$$

Consequentially, an unbiased estimate for $\mathbf{R}$ given an observation $\mathbf{z}_i$ and a distribution over the state $\mathbf{x}_i$ is given by

$$\mathbf{E}\left[(\mathbf{z}_i-\mathbf{H}\boldsymbol{\mu}_i)(\mathbf{z}_i-\mathbf{H}\boldsymbol{\mu}_i)^\top - \mathbf{H}\boldsymbol{\Sigma}_i\mathbf{H}^\top\right] = \mathbf{R} \tag{3.46}$$

The asymptotic results of equations (3.26) and (3.28) relied only on the availability of an unbiased estimator $\mathbf{T}_{\boldsymbol{\theta}}$ for the elements of the covariance matrix; as such, we immediately obtain a consistent estimate of the observation covariance $\mathbf{R}$ at some predictor $\boldsymbol{\phi}$ given just observations $\mathbf{z}_{1:N_D}$ and a multivariate Gaussian distribution over the states $\mathbf{x}_{1:N_D}$.

$$\hat{\mathbf{R}}(\boldsymbol{\phi}) = \frac{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma_{\boldsymbol{\phi}})\left[(\mathbf{z}_i-\mathbf{H}\boldsymbol{\mu}_i)(\mathbf{z}_i-\mathbf{H}\boldsymbol{\mu}_i)^\top - \mathbf{H}\boldsymbol{\Sigma}_i\mathbf{H}^\top\right]}{\sum_{i=1}^{N} k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\sigma_{\boldsymbol{\phi}})} \tag{3.47}$$

The additional term in the numerator accounts for uncertainty added by the lack of perfect knowledge of state.

Because this estimate is consistent, for a sufficiently large data set, the estimate $\hat{\mathbf{R}}(\boldsymbol{\phi})$ will almost surely equal the true covariance $R(\boldsymbol{\phi})$. However, in order to predict observation covariances, we require a distribution over the latent states corresponding to the observations in the data set; in order to evaluate a distribution over the latent states, we require the observation covariances. This circular dependency breaks the tree structure which enables the efficient exact inference of the Kalman filter.

However, we can *approximately* infer the latent states efficiently using the Expectation-Maximization algorithm [14]. Although inferring the states $\mathbf{x}_{1:N_D}$ given just the observations $\mathbf{z}_{1:N_D}$ is difficult, the inference reduces to the Kalman filter given the observation covariances $\mathbf{R}_{1:N_D}$. We also know how to estimate the covariances $\mathbf{R}_{1:N_D}$ given the observations $\mathbf{z}_{1:N_D}$, states $\mathbf{x}_{1:N_D}$, and predictors $\boldsymbol{\phi}_{1:N_D}$. By iterating between these two inference procedures, we may infer both the covariance sequence $\mathbf{R}_{1:N_D}$ and a distribution over the state sequence $\mathbf{x}_{1:N_D}$. This iterative process is guaranteed to converge to a locally maximum likelihood estimate.

To additionally select the maximum likelihood metric and kernel parameters, we augment this two-step iterative process with an optimization step. As before, we express the parameters as a single vector $\boldsymbol{\alpha}$, and note that the best sensor model will maximize the likelihood of the observation sequence.

$$\boldsymbol{\alpha}^{\star} = \arg\max_{\boldsymbol{\alpha}} \prod_{i=1}^{N} p\Big(\mathbf{z}_i | \hat{\mathbf{R}}(\boldsymbol{\phi}_i, \boldsymbol{\alpha})\Big) \tag{3.48}$$

We evaluate this optimization just as in the known-state case, using the current estimate of the state sequence distribution to estimate the covariance sequence.

This procedure, dubbed CELLO-EM, can be summarized in three steps. We choose an initial guess for the hyperparameters, and an initial observation covariance sequence. Often, the initial hyperparameters will be chosen randomly, and the initial covariance sequence will be a constant matrix chosen heuristically for the problem at hand. We then repeatedly cycle through two steps. First, we compute a distribution over the latent state sequence, conditioned on the current estimate of the covariance sequence:

$$p\Big(\mathbf{x}_i | \mathbf{z}_{1:N_D}, \hat{\mathbf{R}}^{(n)}_{1:N_D}\Big) = \mathcal{N}\left(\boldsymbol{\mu}_i^{(n)}, \boldsymbol{\Sigma}_i^{(n)}\right) \quad \forall i \in [1, N_D] \tag{3.49}$$

Then, we compute an observation covariance sequence as a function of the hyperparameter vector $\boldsymbol{\alpha}$,given the current distribution over the latent state sequence.

$$\hat{\mathbf{R}}_i(\boldsymbol{\alpha}) = \frac{\sum_{j \neq i} k(\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|, \boldsymbol{\alpha}) \left[(\mathbf{z}_j - \mathbf{H}\boldsymbol{\mu}_j^{(n)})(\mathbf{z}_j - \mathbf{H}\boldsymbol{\mu}_j^{(n)})^{\top} - \mathbf{H}\boldsymbol{\Sigma}_j^{(n)}\mathbf{H}^{\top}\right]}{\sum_{j \neq i} k(\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|, \boldsymbol{\alpha})} \tag{3.50}$$

Finally, we choose the hyperparameters which maximize the likelihood of the observation sequence,

$$\boldsymbol{\alpha}^{\star}_{n+1} = \arg\max_{\boldsymbol{\alpha}} \prod_{i=1}^{N} p\Big(\mathbf{z}_i|\hat{\mathbf{R}}_i(\boldsymbol{\alpha}), \hat{\mathbf{x}}_i^{(n)}\Big)\, p\Big(\mathbf{x}_i^{(n)}\Big) \tag{3.51}$$

We then use the observation covariance sequence inferred using the maximum likelihood hyperparameters to infer an improved distribution over the state sequence, and continue the cycle until convergence. Once convergence has occurred, we store the maximum likelihood hyperparameters $\boldsymbol{\alpha}^{\star}_{\infty}$ and the parameters of the inferred distribution over the state sequence, $\boldsymbol{\mu}^{(\infty)}_{1:N_D}$ and $\boldsymbol{\Sigma}^{(\infty)}_{1:N_D}$. This information is sufficient to predict the covariances of new samples, using equation (3.47). Algorithm 3 describes the procedure for predicting observation covariance without ground truth; algorithm 4 lists the steps required to learn the hyperparameters $\boldsymbol{\alpha}$ and build the data set, using an external Kalman filter for state inference and gradient descent with random restarts for optimization.

Algorithm 3 Covariance prediction without ground truth

```
Input: Query point $\boldsymbol{\phi}$
Input: Dataset $\mathcal{D} = \{(\mathbf{z}_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \boldsymbol{\phi}_i) \quad \forall i \in [1, N_D]\}$
Input: Parameters $\boldsymbol{\alpha}$ of the kernel function $k(\cdot,\cdot)$
Output: Estimate $\hat{\mathbf{R}}$ of the covariance $\mathbf{R}$ at point $\boldsymbol{\phi}$
  function PredictCovariance($\boldsymbol{\phi}, \mathcal{D}, \boldsymbol{\alpha}$)
      $\hat{\mathbf{R}} \leftarrow \mathbf{0}$, $n \leftarrow 0$
      $\mathcal{S} \leftarrow$Neighbors($\boldsymbol{\phi}$)          ▷ Returns $\mathcal{S} \subset \mathcal{D} : \|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|_{\mathbf{M}} < \sigma \quad \forall i \in \mathcal{S}$
      for $i \in \mathcal{S}$ do
          $\hat{\mathbf{R}} \leftarrow \hat{\mathbf{R}} + k(\boldsymbol{\phi}, \boldsymbol{\phi}_i)\left((\mathbf{z}_i - h(\hat{\mathbf{x}}_i))(\mathbf{z}_i - h(\hat{\mathbf{x}}_i))^{\top} - \mathbf{H}\boldsymbol{\Sigma}_i\mathbf{H}^{\top}\right)$
          $n \leftarrow n + k(\boldsymbol{\phi}, \boldsymbol{\phi}_i)$
      end for
      $\hat{\mathbf{R}} \leftarrow \frac{1}{n}\hat{\mathbf{R}}$
      return $\hat{\mathbf{R}}$
  end function
```

3.6 Bayesian Formulation

Thus far we have tacitly taken the frequentist interpretation of statistics. That is, we have asserted the observation distribution $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ is a multivariate Gaussian with a

Algorithm 4 Covariance estimation using Expectation Maximization (CELLO-EM)

```
Input: Dataset $\mathcal{D} = \{(\mathbf{z}_i, \boldsymbol{\phi}_i) \quad \forall i \in [1, N_D]\}$
Output: Dataset $\mathcal{D}^\star = \{(\mathbf{z}_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \boldsymbol{\phi}_i) \quad \forall i \in [1, N_D]\}$
Output: Kernel parameters $\boldsymbol{\alpha}^\star = \arg\max_{\boldsymbol{\alpha}'} \mathcal{L}(\boldsymbol{\alpha}'|\mathcal{D})$
  function LearnModel($\mathcal{D}$)
      Initialize observation covariance sequence $\hat{\mathbf{R}}_{1:N_D}$
      repeat
          $\{\boldsymbol{\mu}_{1:N_D}, \boldsymbol{\Sigma}_{1:N_D}\} \leftarrow$KalmanFilter($\mathbf{z}_{1:N_D}, \mathbf{R}_{1:N_D}$)
          $\boldsymbol{\alpha} \leftarrow$ MaximizeLikelihood($\mathbf{z}_{1:N_D}, \boldsymbol{\mu}_{1:N_D}, \boldsymbol{\Sigma}_{1:N_D}, \boldsymbol{\phi}_{1:N_D}$)
          for $j \in [1, N_D]$ do
              $\hat{\mathbf{R}}_j \leftarrow$PredictCovariance($\boldsymbol{\phi}_j, \mathcal{D}, \boldsymbol{\alpha}$)
          end for
      until convergence
      return $\boldsymbol{\alpha}^\star, \mathcal{D}^\star = \{\mathbf{z}_{1:N_D}, \boldsymbol{\mu}_{1:N_D}, \boldsymbol{\Sigma}_{1:N_D}, \boldsymbol{\phi}_{1:N_D}\}$
  end function
  function MaximizeLikelihood($\mathbf{z}_{1:N_D}, \boldsymbol{\mu}_{1:N_D}, \boldsymbol{\Sigma}_{1:N_D}, \boldsymbol{\phi}_{1:N_D}$)
      Set learning rate $\boldsymbol{\eta}$
      Set number of restarts $m$
      for $i = [1, m]$ do
          Randomly initialize $\boldsymbol{\alpha}$
          repeat
              for $j \in [1, N_D]$ do
                  $\hat{\mathbf{R}}_j \leftarrow$PredictCovariance($\boldsymbol{\phi}_j, \mathcal{D}, \boldsymbol{\alpha}$)
              end for
              $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \boldsymbol{\eta}\nabla\mathcal{L}\left(\hat{\mathbf{R}}_{1:N_D}, \mathcal{D}\right)$
          until convergence
          if $\mathcal{L}(\boldsymbol{\alpha}|\mathcal{D}) > \mathcal{L}(\boldsymbol{\alpha}^\star|\mathcal{D})$ then
              $\boldsymbol{\alpha}^\star \leftarrow \boldsymbol{\alpha}$
          end if
      end for
      return $\boldsymbol{\alpha}^\star$
  end function
```

fixed covariance $R(\boldsymbol{\phi})$, and have attempted to estimate that covariance function given a set of random independent samples. The resulting algorithm can also be developed within a Bayesian framework. Somewhat surprisingly, the actual calculations are virtually identical; it is purely the interpretation which differs.

We now present the corresponding method for Bayesian inference. Consider the case of Bayesian inference for a fixed covariance. The conjugate prior for the covariance matrix of a multivariate Gaussian distribution is the inverse Wishart distribution.

$$p(\mathbf{R}) = \mathcal{W}^{-1}\left(\boldsymbol{\Psi}, \nu\right) = \frac{|\boldsymbol{\Psi}|^{\frac{\nu}{2}}}{2^{\frac{\nu N_{\mathbf{z}}}{2}}\Gamma_p\left(\frac{\nu}{2}\right)} |\mathbf{R}|^{-\frac{\nu+N_{\mathbf{z}}+1}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Psi}\mathbf{R}^{-1}\right)\right) \tag{3.52}$$

The hyperparameters of this prior are a positive definite *scale matrix* $\boldsymbol{\Psi}$, and a scalar number of *degrees of freedom* $\nu \in \mathbb{R} > N_{\mathbf{z}} - 1$. The scale matrix parameterizes the expected size of the covariance; the expected value of the covariance matrix is

$$\mathbf{E}\left[\mathbf{R}\right] = \frac{\boldsymbol{\Psi}}{\nu - N_{\mathbf{z}} - 1} \tag{3.53}$$

while the maximum likelihood covariance is

$$\mathbf{R}_{\mathrm{ML}} = \frac{\boldsymbol{\Psi}}{\nu + N_{\mathbf{z}} + 1} \tag{3.54}$$

The number of degrees of freedom parameterizes the certainty of the distribution. Large values of ν indicate a high degree of confidence. Commonly, we will choose our prior degrees of freedom as $\nu = N_{\mathbf{z}}$; this is the minimum information non-degenerate inverse Wishart distribution.

Applying Bayes' rule to our observation model, we obtain a recursive Bayesian inference procedure.

$$p(\mathbf{R}|\mathbf{z}) \propto p(\mathbf{z}|\mathbf{R})\, p(\mathbf{R}) \tag{3.55}$$

$$\propto \mathcal{N}\left(h\left(\mathbf{x}\right), \mathbf{R}\right) \mathcal{W}^{-1}\left(\boldsymbol{\Psi}, \nu\right) \tag{3.56}$$

$$= \mathcal{W}^{-1}\left(\boldsymbol{\Psi} + (\mathbf{z} - h\left(\mathbf{x}\right))(\mathbf{z} - h\left(\mathbf{x}\right))^{\top}, \nu + 1\right) \tag{3.57}$$

Given N_D independent samples from $p(\mathbf{z}|\mathbf{R})$, the posterior distribution $p(\mathbf{R}|\mathbf{z}_{1:N_D}, \mathbf{x}_{1:N_D})$ is an inverse Wishart distribution with scale

$$\boldsymbol{\Psi}_{\text{post}} = \boldsymbol{\Psi} + \sum_{i=1}^{N_D} (\mathbf{z}_i - h\left(\mathbf{x}_i\right))(\mathbf{z}_i - h\left(\mathbf{x}_i\right))^{\top} \tag{3.58}$$

and degrees of freedom

$$\nu_{\text{post}} = \nu + N_D \tag{3.59}$$

Note that in the limit of many samples, this distribution converges a delta function around the frequentist estimator.

If the observation covariance is not the same for every sample, the standard procedure for covariance inference no longer applies. By construction, $p(\mathbf{z}|\boldsymbol{\phi})$ is multivariate Gaussian with some covariance $\mathbf{R}$, but the covariance is free to vary with the predictor. Without imposing any additional conditions, a sample may provide information about the covariance only at a single point in predictor space. In order to make inferences about the covariance $\mathbf{R}$ at some point $\boldsymbol{\phi}$ in predictor space where we have no previous sample, we require a distribution over the observation $\mathbf{z}'$ corresponding to some other predictor $\boldsymbol{\phi}'$.

$$p(\mathbf{z}'|\mathbf{R}, \boldsymbol{\phi}, \boldsymbol{\phi}') \tag{3.60}$$

It is not immediately clear what this distribution should be.

In the frequentist algorithm proposed previously, we addressed this by the introduction of the kernel function. These kernel functions restricted the rate at which the covariance could vary. This rate restriction implicitly defines a Lipschitz continuity condition; as the density of data increases, we are able to decrease the size of kernels, increasing our Lipschitz constant and relaxing our continuity condition. We will take a similar approach with the Bayesian derivation.

It is not sensible to describe a continuity condition on the covariance from a Bayesian perspective; the parameters are described as distributions, not deterministic functions. Instead, we restrict the rate at which these distributions may change as the predictors vary. The natural language with which to compare distributions is the Kullback-Leibler, or KL, divergence. The KL divergence of a distribution Q over a random vector $\mathbf{x}$ from a distribution P is defined as

$$D_{\mathrm{KL}}\left(P \| Q\right) = \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) \tag{3.61}$$

We restrict the KL divergence of the distribution at $\boldsymbol{\phi}$ from the distribution at $\boldsymbol{\phi}'$ to be a function of the distance between the predictors $\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|$. Because the distribution at $\boldsymbol{\phi}$ is known, we may write

$$D_{\mathrm{KL}}\left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|\right) = \int \mathrm{d}\mathbf{z}'\, p(\mathbf{z}'|\mathbf{R}, \boldsymbol{\phi}, \boldsymbol{\phi}') \log\left(\frac{p(\mathbf{z}'|\mathbf{R}, \boldsymbol{\phi}, \boldsymbol{\phi}')}{\mathcal{N}\left(\mathbf{0}, \mathbf{R}\right)}\right) \tag{3.62}$$

The only constraint on $D_{\mathrm{KL}}\left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|\right)$ is that it must be zero if the distance between predictors is zero. Beyond that we are free to choose $D_{\mathrm{KL}}\left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|\right)$ as we see fit, just as we were free to choose kernel functions in the frequentist interpretation.

Many distributions will satisfy this relation for a particular choice of $D_{\mathrm{KL}}\left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|\right)$. The principle of maximum entropy suggests that we should choose among this set of distributions the one that maximizes its information-theoretic entropy. That is, we should choose

$$p(\mathbf{z}'|\mathbf{R}, \boldsymbol{\phi}, \boldsymbol{\phi}') = \arg\max_{p} \int \mathrm{d}\mathbf{z}'\, -p(\mathbf{z}') \log p(\mathbf{z}') \tag{3.63}$$

subject to

$$0 = \int \mathrm{d}\mathbf{z}'\, p(\mathbf{z}'|\mathbf{R}, \boldsymbol{\phi}, \boldsymbol{\phi}') \left(D_{\mathrm{KL}}\left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|\right) - \log\left(\frac{p(\mathbf{z}'|\mathbf{R}, \boldsymbol{\phi}, \boldsymbol{\phi}')}{\mathcal{N}\left(\mathbf{0}, \mathbf{R}\right)}\right)\right) \tag{3.64}$$

We employ variational calculus to determine the distribution $p(\mathbf{z}'|\mathbf{R}, \boldsymbol{\phi}, \boldsymbol{\phi}')$; a derivation is available in appendix B. The maximum entropy distribution satisfying equation (3.62) is

$$p(\mathbf{z}'|\mathbf{R},\boldsymbol{\phi},\boldsymbol{\phi}') \propto \mathcal{N}\left(\mathbf{0},\mathbf{R}\right)^{k(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|)} \tag{3.65}$$

The function $k(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|)$ is a transformation of $D_{\mathrm{KL}}\left(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|\right)$ with the property that $k(0) = 1$ and $\lim_{\rho\to\infty} k(\rho) = 0$. Because we were free to choose the function $D_{\mathrm{KL}}\left(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|\right)$ to suit our needs, we may instead directly choose $k(\rho)$. We can then freely identify $k(\rho)$ with the kernel functions $k(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|,\boldsymbol{\alpha})$ used previously.

Using this new distribution for $p(\mathbf{z}'|\mathbf{R},\boldsymbol{\phi},\boldsymbol{\phi}')$ and applying Bayes' rule yields a remarkably simple posterior. We assume an inverse Wishart prior over the covariance at predictor $\boldsymbol{\phi}$, with scale matrix $\boldsymbol{\Psi}$ and degrees of freedom ν.

$$\begin{aligned}
p(\mathbf{R}|\mathbf{z}',\boldsymbol{\phi},\boldsymbol{\phi}') &\propto p(\mathbf{z}'|\mathbf{R},\boldsymbol{\phi},\boldsymbol{\phi}')\,p(\mathbf{R},\boldsymbol{\phi}) && (3.66)\\
&\propto \mathcal{N}\left(\mathbf{z}'|h\left(\mathbf{x}\right),\mathbf{R}\right)^{k(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|,\boldsymbol{\alpha})}\,\mathcal{W}^{-1}\left(\boldsymbol{\Psi},\nu\right) && (3.67)\\
&= \mathcal{W}^{-1}\left(\boldsymbol{\Psi}+k(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|,\boldsymbol{\alpha})(\mathbf{z}'-h\left(\mathbf{x}\right))(\mathbf{z}'-h\left(\mathbf{x}\right))^\top,\nu+k(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|,\boldsymbol{\alpha})\right) && (3.68)
\end{aligned}$$

Given a data set $\mathcal{D}$ of N_D independent samples $(\mathbf{z},\mathbf{x},\boldsymbol{\phi})$, the posterior distribution $p(\mathbf{R}|\mathcal{D},\boldsymbol{\phi})$ is an inverse Wishart distribution with scale

$$\boldsymbol{\Psi}_{\text{post}} = \boldsymbol{\Psi}+\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\boldsymbol{\alpha})(\mathbf{z}_i-h\left(\mathbf{x}_i\right))(\mathbf{z}_i-h\left(\mathbf{x}_i\right))^\top \tag{3.69}$$

and degrees of freedom

$$\nu_{\text{post}} = \nu+\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_i\|,\boldsymbol{\alpha}) \tag{3.70}$$

If the states $\mathbf{x}_{1:N_D}$ are unavailable, they may be inferred using an expectation-maximization procedure identical to algorithm 4. Note that in the limit of many samples, both the expected covariance matrix and the maximum likelihood covariance matrix converge to the estimate provided generated by CELLO (equation (3.22)).

$$\lim_{N_D \to \infty} \mathbf{E}\left[\mathbf{R}\right] = \lim_{N_D \to \infty} \mathbf{R}_{\mathrm{ML}} = \frac{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|, \boldsymbol{\alpha}) \mathbf{v}_i \mathbf{v}_i^\top}{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|, \boldsymbol{\alpha})} \tag{3.71}$$

Note also that we exactly recover the frequentist estimator, even for finite data, if our prior distribution is the degenerate inverse Wishart distribution with scale matrix $\mathbf{\Psi} = \mathbf{0}$ and degrees of freedom $\nu = 0$. This prior asserts the belief that the covariance is zero—that is, that our observations are perfectly accurate—but also implies minimal certainty in that belief.

The Bayesian variation of CELLO is virtually identical to the frequentist algorithm; in practice, the only difference is the initialization of the outer product and kernel sums to a nonzero value. This has the practical advantage of increasing robustness; the Bayesian algorithm produces a reasonable estimate even if the query point is in a region with very low data density. Moreover, it has been shown by Agamennoni et al. [1] that marginalizing the covariance of an inverse Wishart-Normal observation model yields a filter which is more robust to outliers than the standard Kalman filter. This suggests a promising route towards making use of the information CELLO provides about the uncertainty of its covariance estimates.

Chapter 4

Simulation Results

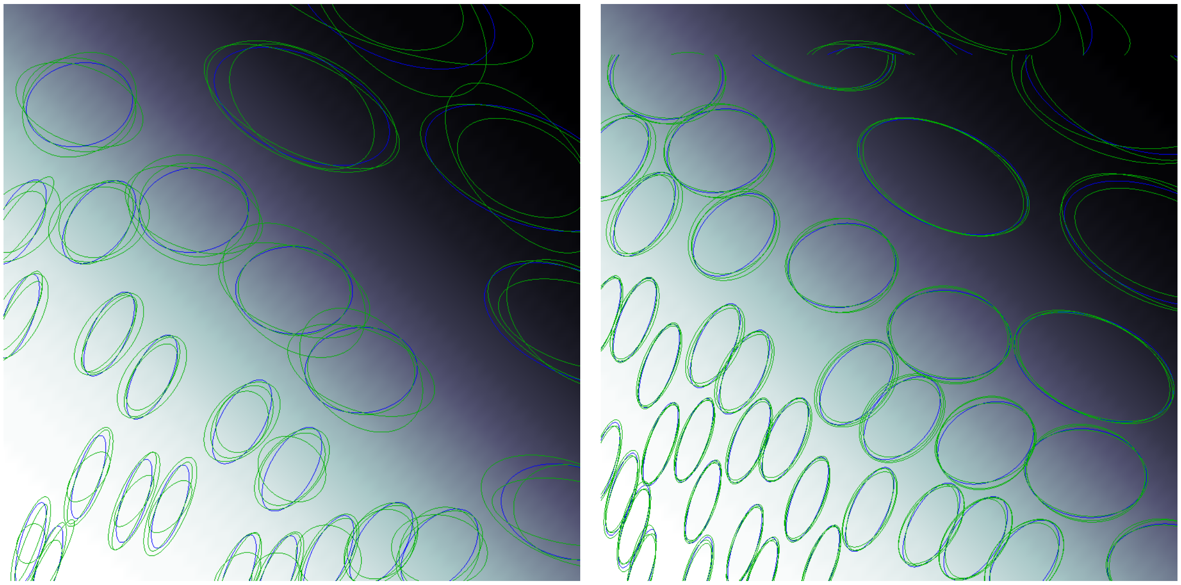
To evaluate CELLO and its derivatives, I developed several simulation environments. This allowed the validation of performance with perfect access to ground truth, perfect models, and perfect repeatability, three things which are difficult to attain in real data.

4.1 Dark Room

The first simulation experiment conducted considered a fictional robot taking position measurements in a room of varying brightness. The fictional position sensor was constructed to perform well in the light, but poorly in the darkness; the robot navigated the room, and compared sensor measurements to ground truth values at many locations. The difference between observed position and true position was recorded as an error vector $\mathbf{v}_i$.

Although the observation covariance was a function of the brightness, I used the true position as a predictor vector $\boldsymbol{\phi}_i$. This choice is atypical, but valid, as the brightness was itself a function of the position. Since the function mapping position to brightness was known, choosing the position as a predictor allowed assessment of the optimality of the learned hyperparameters, thereby permitting validation of the learning process.

Pairs of predictor vectors $\boldsymbol{\phi}_i$ and noise vectors $\mathbf{v}$ provided the data set $\mathcal{D}$ required by algorithm 2. I used the scaled Euclidean metric discussed in chapter 3, and learned



(a) *100 samples*

(b) *1000 samples*

Figure 4-1: *Two-dimensional covariance learning. The blue ellipses are true measurement covariances drawn from the data set; the green ellipses are the optimized estimated covariances, with results drawn from several independent data sets superimposed. In figure 4-1a, 100 samples were provided to CELLO for learning and prediction; in figure 4-1b, 1000 samples were provided. For both figures a random set of query points was chosen for illustrative purposes; the points were selected so that the covariances would not overlap, to improve visibility. In each case, the estimates appear unbiased, but the variance of the predicted covariances is visibly smaller when more data is available.*

the hyperparameters as described in algorithm 2. I tested the learning process several times, on independent data sets of varying sizes. The results of the learning process are presented in figure 4-1, when (figure 4-1a) 100 and (figure 4-1b) 1000 samples were used for training. Visually, the increased availability of samples dramatically reduces the variance of the estimates; Figure 4-2 presents the mean squared error between the predicted and true covariances, as a function of the number of samples provided to the learning algorithm; the error rapidly decreases as the number of samples increases.

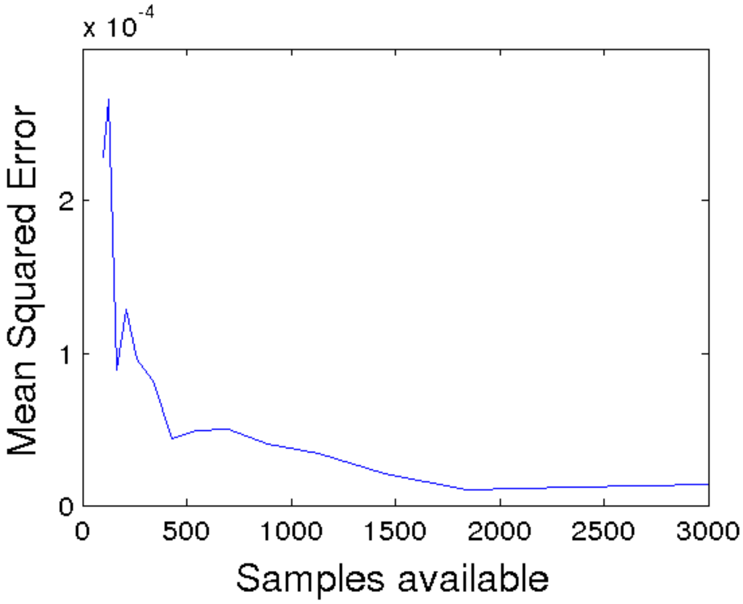


Figure 4-2: The mean squared error between the predicted and true covariances decreases as the number of samples available decreases.

4.2 Random Walk

To evaluate CELLO-EM, I augmented the previous simulation with a process model, *viz.*

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{w}_t \tag{4.1}$$

where the state transition matrix $\mathbf{F}$ was chosen to have ellipses, with eccentricity $\frac{\sqrt{3}}{2}$, for its nominal steady state trajectories:

$$\mathbf{F} = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \exp\left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} s \right) \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix}^{-1} \tag{4.2}$$

The constant factor s defined the walking speed. For the experiments presented here, I set $s = 0.1$. The process noise $\mathbf{w}$ was drawn from a multivariate Gaussian distribution with zero mean and covariance $\mathbf{Q} = 5 \times 10^{-4}\mathbb{1}$. The full process model was a random walk, with a general clockwise motion.

This walk took the robot through areas of varying brightness. After each step, the robot took a position measurement, which was corrupted by Gaussian noise.

As before, the fictional position sensor performed well in the light, but poorly in the darkness; as the robot wandered around the room, it recorded both a noisy measurement of position and a predictor vector consisting of the observed brightness, and the direction of the nearest light source. An optimal filter for this system would trust its process model in the dark areas, when measurements are very noisy; in brighter areas, it would favor the measurements. A fixed measurement covariance could not perform this trade-off, and must either place too much confidence in the measurements when in the dark areas, or too little in the bright areas.

This system was, by construction, a linear Gaussian system, and therefore if the measurement covariances were known it would be possible to exactly solve for the least-squares optimal solution using a standard Kalman filter. I use this exact solution as a baseline for comparison in all simulation experiments in this domain; it represents the best possible estimate given the data available, and provides a lower bound on the possible filter error.

I processed a simulated dataset as described in algorithm 4, initializing the covariance to a fixed estimate and iteratively improving that estimate through cycles of Kalman filtering and CELLO maximization. I then processed the same data using the fixed-covariance expectation maximization; this generated the maximum likelihood estimate of the covariance and as such the best possible performance from a fixed-covariance Gaussian measurement model. The resulting mean squared error for each algorithm is compared in figure 4-3. Both the error resulting from filters using CELLO and from filters using the fixed covariance rapidly converge to a minimum; however, the fixed-covariance model remains suboptimal even after hundreds of iterations. Using covariances estimated by CELLO reduced the mean squared error by a factor of two, relative to an identical Kalman using the optimal fixed covariance model.

The trajectories estimated using covariances generated by CELLO, by the fixed covariance chosen by expectation maximization, and by the fixed covariance used for initialization, are presented in figure 4-5, and compared to the optimal estimate. In the dark, both CELLO and the fixed covariance estimate perform similarly, trusting

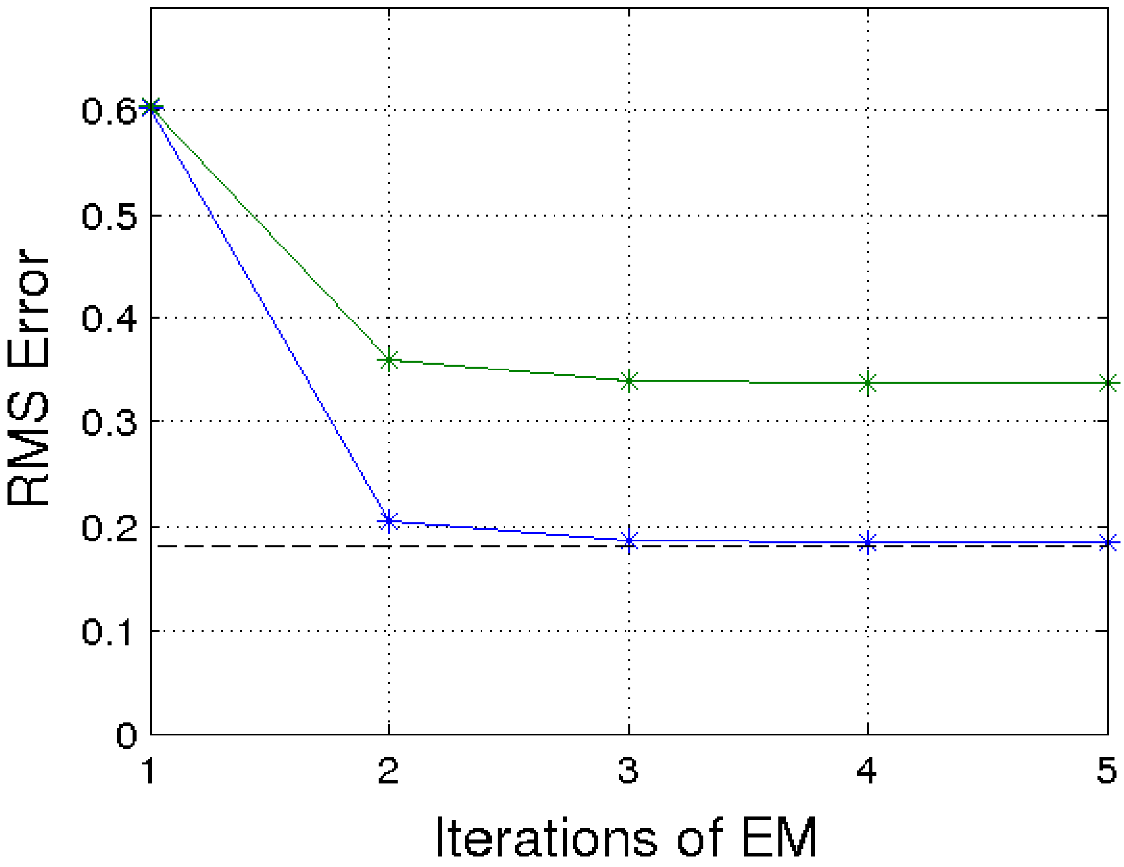


Figure 4-3: *Root mean squared error between the estimated and true trajectories, using covariances generated by CELLO and using a fixed covariance chosen by expectation maximization. Both algorithms converged after only a few iterations, but due to variations in the true covariance, the fixed covariance model failed to achieve optimality. CELLO converged to the optimum, and resulted in half the mean error.*

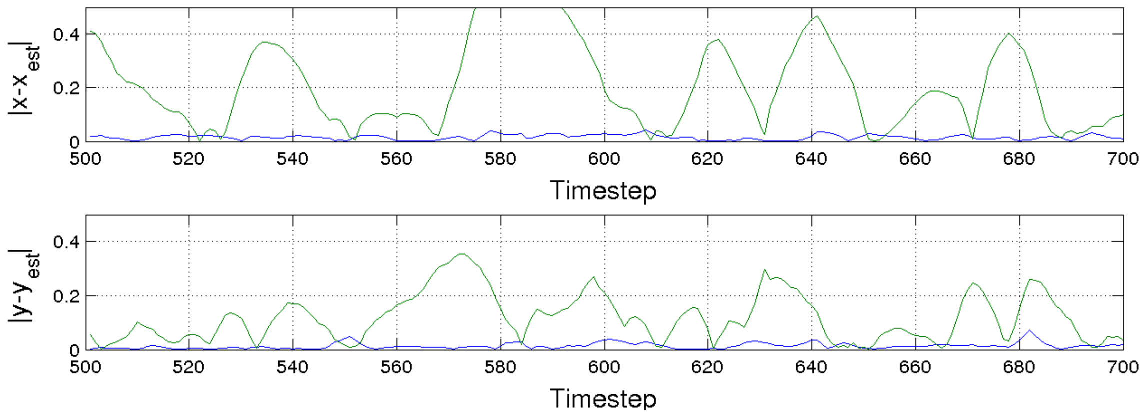


Figure 4-4: *Comparison of estimation error magnitude using covariances predicted by CELLO (blue) and fixed covariances chosen by expectation maximization (green) in each dimension. Error is relative to the optimal estimate; note that when there is little information available, all systems perform comparably. When information is available, however, the fixed covariance model fails to incorporate it, leading to degraded performance relative to the optimum. The estimator using covariances chosen by CELLO performs comparably to the optimal estimator regardless of information quality.*

the dynamics model more than the sensor measurements, resulting in very smooth trajectories. In the light regions, however, CELLO is able to recover the random motions of the vehicle from the sensor data, while the fixed covariance estimator remains smooth, placing too much confidence in the system dynamics model over the sensor measurements. These trends are reflected in the absolute errors for each estimate, presented in figure 4-4. In the dark areas, the estimator using CELLO, the estimator using fixed covariances, and the optimal estimator using the true covariances all performed comparably. However, the performance of the fixed covariance estimate fails to improve in the regions of high information, resulting in large peaks in the error.

4.3 Scan-Matching

Consider the problem of laser scan-matching odometry: given two overlapping scans $\{\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2\}$ from a laser range finder, compute a distribution over possible translations $\boldsymbol{\Delta}$ and rotations $\mathbf{R}$, $p(\boldsymbol{\Delta}, \mathbf{R}|\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2)$. If the scans are sequential and separated by a known time Δt, then $\boldsymbol{\Delta} = \mathbf{v}\Delta t$, where $\mathbf{v}$ is the vehicle velocity; the rotation between frames can be likewise related to the angular velocity. If two scans can be aligned,

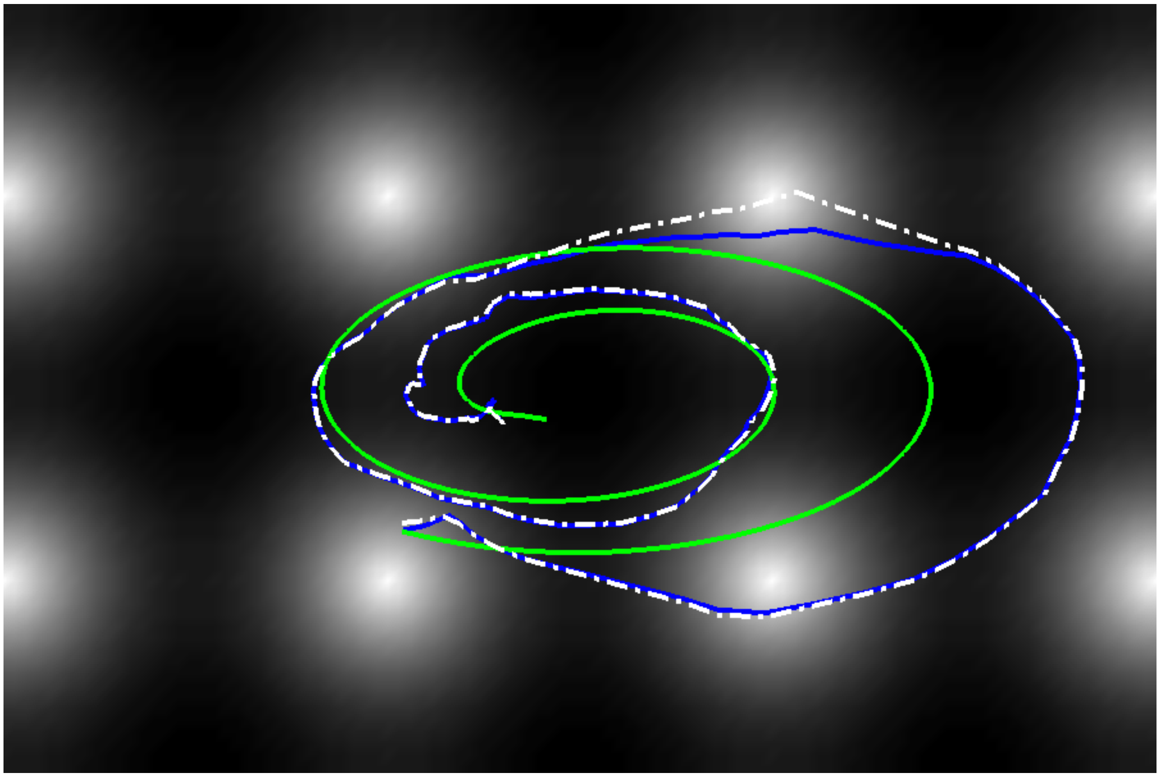
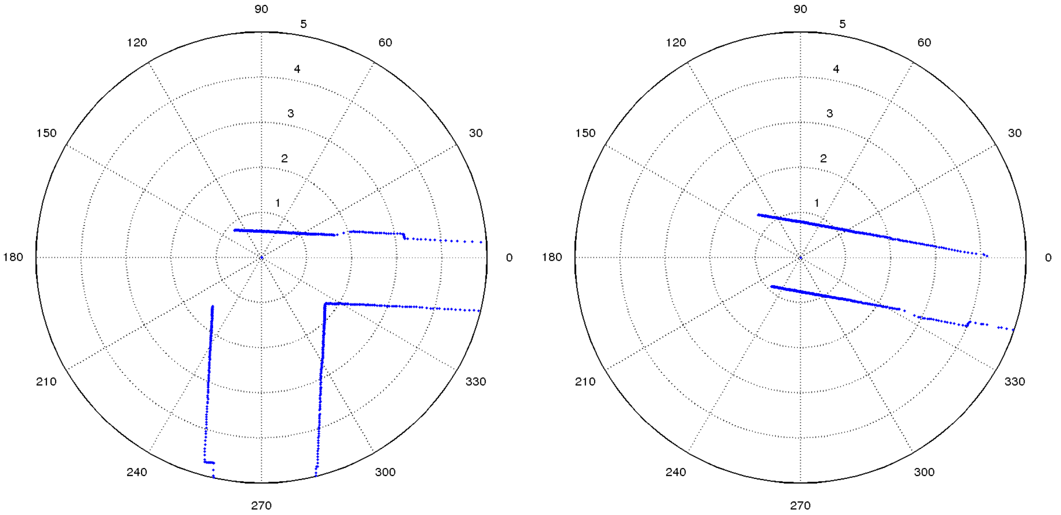


Figure 4-5: *Estimated trajectory using covariances predicted by CELLO (blue), fixed covariances chosen by expectation maximization (green), and the true measurement covariances (white). Trajectories are presented in front of the light field indicating sensor quality: the sensor degrades in the dark. Note that using the true covariances gives an upper bound on estimator performance; CELLO nearly achieves this upper bound, while the fixed-covariance model fails to do so, especially in the dark areas.*



(a) Corner

(b) Hallway

Figure 4-6: *Sequential laser scans can be matched to estimate the velocity of a robot, but environmental degeneracy can lead to less information in some directions. Near a corner, the robot can accurately estimate changes in both heading and each position axis; in a hallway, the robot can only estimate motion in the direction of the walls.*

then it is possible to infer the vehicle velocity and rotation rate from the optimal transform.

There are many ways of inferring, or approximating, this distribution. I computed the transform between scans using the algorithm described by Bachrach et al. [5]. This algorithm creates a local map by extracting contours from a recent history of scans, and formulates the scan-matching problem as an optimization procedure, choosing the transform which maximizes the likelihood of the observed scan given the recent map. However, if the map is ambiguous, such as in a hallway, the likelihood function will not be well-conditioned, and the maximum likelihood transform will be highly sensitive to measurement noise. Consider the sample scans presented in figure 1-2, and reproduced in figure 4-6. When there is structure in both directions, as in figure 4-6a, matching scans reliably yields information about both the velocity and yaw rate of a vehicle. However, when the structure is ambiguous, as in figure 4-6b, motion cannot be detected in one direction. Depending on the method used to compute the translation and rotation between scans, this can lead to spurious observations of motion along the axis of the hallway.

In order to reject these spurious observations of motion and improve estimation performance, I first simulated a sequence of two dimensional range scans in an idealized infinite hallway. This simulated LIDAR unit had finite range; additionally, the simulated range measurements were corrupted by a small amount of noise. For each pair of scans, I used the scan-matching odometry system of Bachrach et al. to generate an estimate of the translation and rotation between scans; dividing by the time between scans gave an estimate of velocity. Using the same pair of scans, I generated a predictor vector composed of two histograms: a count the number of returned measurements in each of ten bins of angles, and a count of angles of the line segment connecting each sequential scanned point. These were both chosen to indicate the presence of walls.

Because this was done in simulation, the ground truth velocity at each time step was available, allowing the computation of error between the predicted and measured velocity observations. Together, those error vectors and the predictor vector composed of histograms formed a dataset $\mathcal{D}$, which was provided to CELLO. The 2σ covariance ellipse predicted by CELLO after learning hyperparameters are drawn in figure 4-7 at several locations in a hallway. Note that these are the covariances of observations, but are drawn in the state space, to illustrate the alignment of the ellipses. The predictions consistently aligned the covariance ellipse with the hallway, regardless of the robot orientation.

I also simulated accelerometer data, and used the predicted covariances to incorporate the scan matching observations into an unscented Kalman filter, A comparison of filter performance using (figure 4-8a) fixed covariances and (figure 4-8b) learned covariances indicates the weaknesses of a fixed covariance scheme; the filter underestimates its uncertainty in the longitudinal direction but grossly overestimates it in the transverse direction. The result is reflected in the distribution of estimated trajectories (solid lines). The learned covariance scheme does a much better job restricting its estimates to the interior of the hallway.

Because the noise properties and simulated environment were chosen arbitrarily, these results cannot be used to draw conclusions about the impact of CELLO on

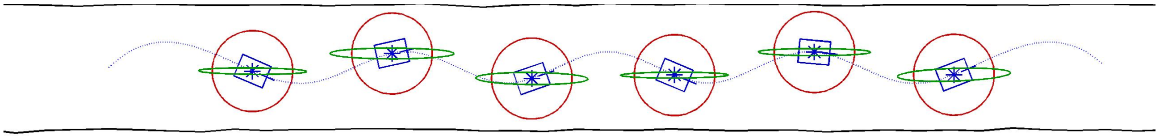
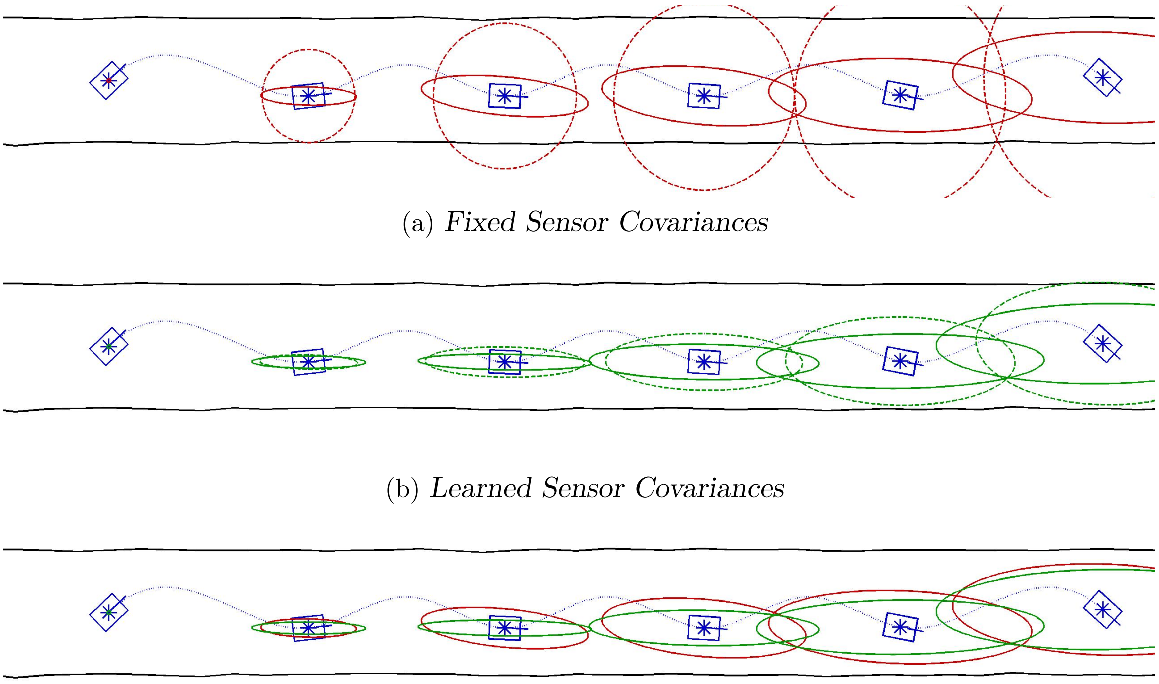


Figure 4-7: *Learned measurement covariances (green) for a scan matching algorithm in a hallway, compared to fixed measurement covariances (red) taken as the empirical covariance of the available sample data. Measurements are displacements between successive scans, consequently they have the same domain as the position variables; covariance ellipses are presented at arbitrary scale, hence it is only the orientation and eccentricity of these ellipses that is relevant. The learned covariances consistently indicate a large measurement variation in the longitudinal direction, and a small variation in the transverse direction, reflecting the availability of information in only one direction. Note the learning was done using predictor features taken purely from the scan, and not including the position or orientation of the robot—the rotation of the ellipse is an emergent behavior.*

real-world performance. However, the desired properties emerged from the simulated experiment. CELLO was able to learn that little information was available in one direction, and able to consistently learn in which direction information was unavailable. This was done without any prior encoding of these properties, using just a simple model of the desired domain.



(a) *Fixed Sensor Covariances*

(b) *Learned Sensor Covariances*

(c) *Comparison of Simulated Covariances*

Figure 4-8: *Comparison of filter performance using learned and fixed measurement covariances. Note that in contrast to figure 4-7, these covariances are state covariances. Covariances are drawn as 95% confidence margin ellipses at fixed time intervals. Each estimated covariance (dashed lines) is the mean filter covariances for a given time interval i,* $\mathbf{E}\left[\boldsymbol{\Sigma}_i\right]$, *over 250 simulated trials. Each simulated covariance (solid lines) is the covariance of the estimated trajectories for the time interval i,* $\operatorname{cov}\left(\boldsymbol{\mu}_i\right)$, *again over 250 simulations. Filtering was done using an unscented Kalman filter with noisy accelerometer data and the scan-matching output for measurements. Note the distortion of the ellipse in the learned case, reflecting increased uncertainty in the longitudinal direction of the corridor. Note also that the fixed covariance scheme underestimates its uncertainty in the longitudinal direction.*

Chapter 5

Experimental Results

In this chapter, I present the results of several experiments with real sensors and real data. I experimentally verify the impact of a predictive covariance scheme on state estimation using indirect observations of state in ambiguous environments, and I compare the results of filtering using covariances predicted by CELLO to filtering using competing algorithms.

5.1 Optical Flow

Cameras and vision systems play a prominent role in modern robotics; machine vision is an entire field unto itself. The subproblem of *visual odometry*—the problem of estimating camera motion from a sequence of images—is a useful way to utilize visual systems when computational resources are at a premium. Solutions which infer the structure of the environment are computationally expensive, and if a detailed model of the environment is not necessary for a robot to achieve its objectives, it is desirable to avoid this expense. Given a static environment and an image stream where motion between successive frames is small, it is possible to track just the apparent motion of the image and neglect the structure of the environment. Two seminal approaches are the work of Horn and Schunck [21], which calculates the apparent motion of each pixel in an image, as well as that of Lucas and Kanade [31], which extracts a single mean translation from sequential image pairs. Significant progress has since been made; Beauchemin and Barron [9] present a survey.

The connection between apparent image motion and camera motion is complex. I refer to the image returned by a camera at time t as an intensity field $I(\mathbf{v}, t)$, defined in image coordinates $\mathbf{v} \in \mathbb{R}^2$. I denote the camera heading as a unit vector $\hat{\mathbf{u}} \in \mathbb{R}^3$, the camera location as a vector $\mathbf{x} \in \mathbb{R}^3$, and the camera focal length as $f \in \mathbb{R}$; I define a projection matrix $\mathbf{P} \in \mathbb{R}^{2\times 3}$ which maps three-dimensional positions in a frame fixed to the camera to image coordinates. Any point in space $\boldsymbol{\delta} \in \mathbb{R}^3$, where $\boldsymbol{\delta} = \mathbf{0}$ corresponds to the camera position $\mathbf{x}$, can then be projected into the camera frame.

$$\mathbf{v} = -f\mathbf{P}\frac{(\mathbb{1} - \hat{\mathbf{u}}\hat{\mathbf{u}}^\top)\boldsymbol{\delta}}{\hat{\mathbf{u}}^\top\boldsymbol{\delta}} \tag{5.1}$$

If the scene is static, but the camera is permitted to move, fixed points in the inertial frame will appear to move in the camera frame. Taking the derivative of equation (5.1) parameterizes this motion in terms of the linear velocity $\dot{\mathbf{x}}$ and angular velocity $\boldsymbol{\Omega}$ of the camera.

$$\dot{\mathbf{v}} = f\mathbf{P}\frac{\hat{\mathbf{u}}^\top\boldsymbol{\delta}\mathbb{1} - \boldsymbol{\delta}\hat{\mathbf{u}}^\top}{(\hat{\mathbf{u}}^\top\boldsymbol{\delta})^2}(\dot{\mathbf{x}} + \boldsymbol{\Omega}\times\boldsymbol{\delta}) \tag{5.2}$$

This equation is a standard result of image theory, and I omit a derivation. Equation (5.2) can be rewritten in terms of the point depth $\rho = \hat{\mathbf{u}}^\top\boldsymbol{\delta}$ and the image coordinate $\mathbf{v}$.

$$\dot{\mathbf{v}} = \left(\frac{f}{\rho}\right)\mathbf{P}\dot{\mathbf{x}} + \frac{\hat{\mathbf{u}}^\top\dot{\mathbf{x}}}{\rho}\mathbf{v} + f(\mathbb{1} - \frac{1}{f^2}\mathbf{v}\mathbf{v}^\top)\mathbf{P}(\boldsymbol{\Omega}\times\hat{\mathbf{u}}) + (\boldsymbol{\Omega}^\top\hat{\mathbf{u}})\boldsymbol{\sigma}\mathbf{v} \tag{5.3}$$

The matrix $\boldsymbol{\sigma} = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$ is the unit-determinant basis for $\mathfrak{so}(2)$, so that $\boldsymbol{\sigma}\mathbf{v}$ is always perpendicular to and of the same length as $\mathbf{v}$. Note that the apparent motion of any point in image space decomposes into four components, corresponding to translation and rotation in the direction of and perpendicular to the axis of the camera.

Integrating equation (5.3) across the visible image gives

$$\langle\dot{\mathbf{v}}\rangle = f\left\langle\frac{1}{\rho}\right\rangle\mathbf{P}\dot{\mathbf{x}} + f(\mathbb{1} - \left\langle\mathbf{v}\mathbf{v}^\top\right\rangle)\mathbf{P}(\boldsymbol{\Omega}\times\hat{\mathbf{u}}) \tag{5.4}$$

where $\langle\cdot\rangle$ denotes the average. This average flow velocity can be computed without detecting features; if two sequential images $I(\mathbf{v};t)$ and $I(\mathbf{v};t+\delta t)$ can be aligned such that $I(\mathbf{v};t) = I(\mathbf{v}+\boldsymbol{\Delta};t+\delta t)$, then the average velocity is given by $\langle \dot{\mathbf{v}} \rangle \, \delta t = \boldsymbol{\Delta}$. Similarly, if sequential images can be aligned by $I(\mathbf{v};t) = I(\mathbf{T}\mathbf{v};t+\delta t)$, where $\mathbf{T}$ is a linear transform consisting of a rotation and a scale,

$$\mathbf{T} = \exp(s) \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \tag{5.5}$$

then the average scale shift s can be computed as

$$s = \delta t \left\langle \frac{\mathbf{v}}{\mathbf{v}^\top \mathbf{v}} \cdot \dot{\mathbf{v}} \right\rangle = \delta t \left\langle \frac{1}{\rho} \right\rangle \hat{\mathbf{u}}^\top \dot{\mathbf{x}} \tag{5.6}$$

and the average rotation θ is

$$\theta = \delta t \left\langle \frac{\mathbf{v}}{\mathbf{v}^\top \mathbf{v}} \times \dot{\mathbf{v}} \right\rangle = \delta t (\boldsymbol{\Omega}^\top \hat{\mathbf{u}}). \tag{5.7}$$

The four relations of equations (5.4), (5.6) and (5.7) may be aggregated into a single affine transform relating successive images. The parameters of this affine transform may be efficiently computed through the use of the Fourier-Mellin transform.[1] This procedure was first described by De Castro and Morandi [13], and was applied to the problem of motion estimation by Goecke et al. [17]. For convenience, I briefly outline the method here.

I assume sequential images can be exactly aligned by an affine transform: $I(\mathbf{v};t) = I(\mathbf{T}\mathbf{v}+\boldsymbol{\Delta};t+\delta t)$. The Fourier transform of this equation yields

$$\mathcal{F}\{I\}(\tilde{\mathbf{v}}) = \mathcal{F}\{I'\}(\mathbf{T}\tilde{\mathbf{v}}) \exp(j\boldsymbol{\Delta}^\top \tilde{\mathbf{v}}) \tag{5.8}$$

where I use the notation I' to refer to the image at time $t+\delta t$. I have exploited the

[1] Note that although it is possible to disambiguate between arbitrary rotation and translation by additionally considering non-affine coordinate transforms, doing so precludes the use of the efficient Fourier-Mellin procedure outlined here. In addition, in addition, in the configuration used in our experiments, the two unobservable parameters are the pitch and roll rate, which are directly observable using a gyroscope. This renders disambiguation unnecessary.

linearity of the Fourier transform, as well as the fact that a translation in coordinate space is a phase shift in frequency space, to write the transformed image in this form. Equating the magnitudes of both Fourier transforms eliminates the dependency on the translation $\mathbf{\Delta}$.

$$\|\mathcal{F}\{I\}(\tilde{\mathbf{v}})\| = \|\mathcal{F}\{I'\}(\mathbf{T}\tilde{\mathbf{v}})\| \tag{5.9}$$

Next, I change to log-polar coordinates,

$$\tilde{\mathbf{v}} = \exp(\xi_1)\begin{pmatrix}\cos(\xi_2)\\ \sin(\xi_2)\end{pmatrix} \tag{5.10}$$

and define the magnitude image $M(\tilde{\xi_1},\tilde{\xi_2}) = \|\mathcal{F}\{I\}(\tilde{\mathbf{v}})\|$. The new coordinate system was chosen so that a rotation and scale in the original coordinates will be a translation in the new coordinates.

$$\mathbf{T}\tilde{\mathbf{v}} = \exp(\xi_1 + s)\begin{pmatrix}\cos(\xi_2+\theta)\\ \sin(\xi_2+\theta)\end{pmatrix} \tag{5.11}$$

Taking the Fourier transform once more, the translation in log-polar space again becomes a phase shift.

$$\mathcal{F}\{M\}(\tilde{\xi_1},\tilde{\xi_2}) = \mathcal{F}\{M'\}(\tilde{\xi_1},\tilde{\xi_2})\exp(js\tilde{\xi_1} + j\theta\tilde{\xi_2}) \tag{5.12}$$

The parameters s and θ can now be uniquely identified as the parameters that maximize the cross-correlation between the magnitude images M and M'.

$$\{s,\theta\} = \arg\max_{s,\theta}\mathcal{F}^{-1}\{\mathcal{F}\{M\}^{\dagger}\mathcal{F}\{M'\}\}(s,\theta) \tag{5.13}$$

Given the scale and rotation, we can compute the translation in much the same way; we compute a rotated, scaled image $I_{\mathbf{T}}$ and choose the translation that maximizes cross correlation.

$$\mathbf{\Delta} = \arg\max_{\mathbf{\Delta}}\mathcal{F}^{-1}\{\mathcal{F}\{I\}^{\dagger}\mathcal{F}\{I_{\mathbf{T}}\}\}(\mathbf{\Delta}) \tag{5.14}$$

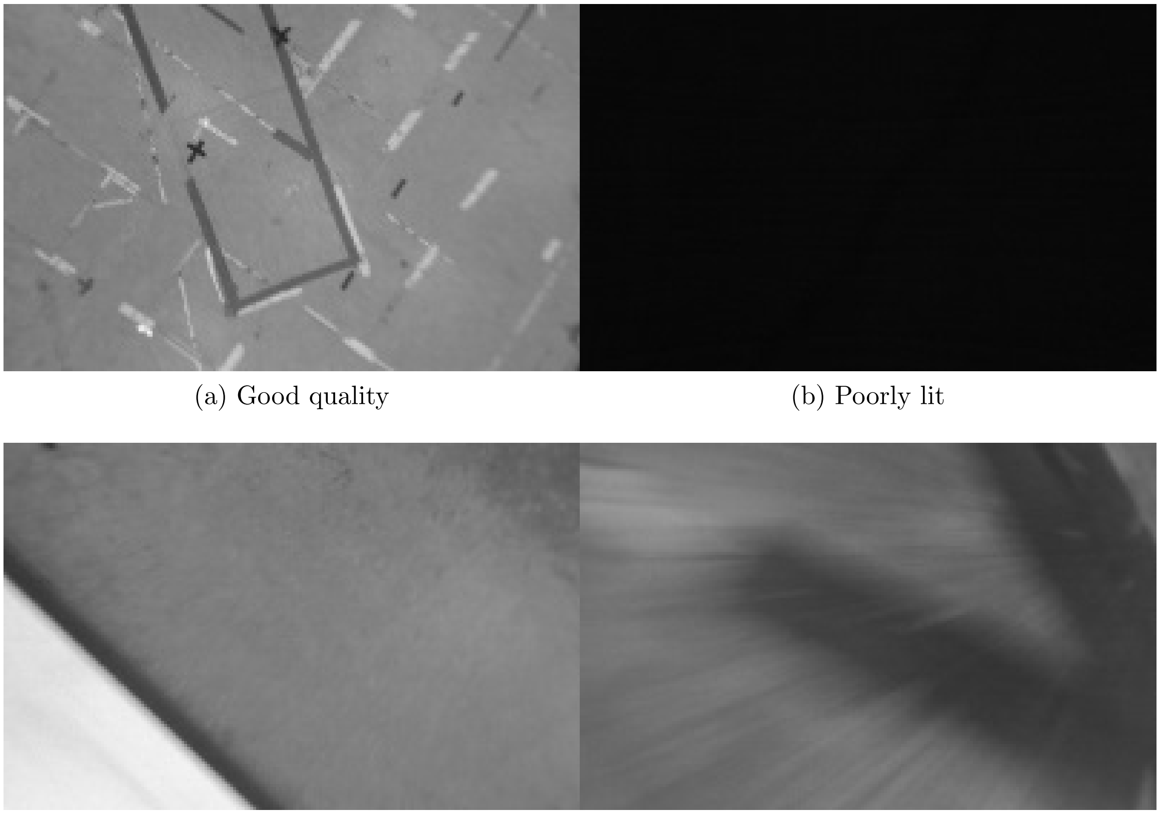
The full procedure requires five Fourier transforms, two changes of coordinates, and two inverse Fourier transforms. However, two Fourier transforms and one coordinate transform may be reused for successive images. Using fast Fourier transforms and bilinear interpolation for the changes of coordinates makes this procedure highly efficient; it is also highly robust to noise.

I implemented this procedure, and computed the transform between sequential images from a downward facing camera on a quadrotor helicopter. In an indoor environment, where the floor is very nearly flat, the quantity $\left\langle \frac{1}{\rho} \right\rangle$ is just the inverse of the vehicle height; the rotation angle gives the vehicle yaw rate, while the scale shift gives vertical velocity and the translation gives horizontal velocity. Although the raw sensor measurement—the camera image—is high-dimensional and environment-dependent, the Fourier-Mellin registration procedure gives a low-dimensional observation which is a simple function of the vehicle state.

$$\mathbf{z} = h\left(\mathbf{x}\right) = \begin{pmatrix} \frac{1}{z}\dot{\mathbf{\Delta}} \\ \mathbf{\Omega}_z \end{pmatrix} \tag{5.15}$$

Here $\dot{\mathbf{\Delta}}$ is the velocity of the vehicle, z is its height above the floor, and $\mathbf{\Omega}_z$ is the yaw component of its angular velocity. By fusing these estimates with an IMU, the full twelve-dimensional state of the vehicle can be estimated.

Naïvely fusing the data in an extended Kalman filter gives very poor results, however. The singularity at $z = 0$ implies that the observations will be highly inaccurate near the ground; even small uncertainty in the estimated height will cause large variations in the inferred velocity. In addition, the algorithm relies on the presence of structure in the images to perform its alignment. In low-texture environments, in darkness, or when images are blurred by rapid motion, the image registration process is impaired and motion estimates degrade. If there was structure only in one direction—for instance, if the camera was looking at a field of parallel stripes—the algorithm could only provide information about the direction with structure. Examples of these low quality measurements are given in figure 5-1, along with a sample



(a) Good quality

(b) Poorly lit

(c) Ambiguous structure

(d) Blurred by motion

Figure 5-1: *The quality of images collected by a downward facing camera varies dramatically depending on the vehicle state and on the environment.*

high-quality image.

CELLO provides a way of predicting the covariance of the observations. Fourteen predictor features were chosen to capture the presence and directionality of structure in the image. These are summarized in section 5.1.

The vehicle was flown in a motion capture room; the motion capture system provides extremely accurate observations of the vehicle state $\mathbf{x}$. These observations were treated as ground truth, and this state information was used to predict the optical flow observations $\{h\left(\mathbf{x}\right)\}$. The transforms obtained from each sequential image pair and the predicted observations from the motion capture system were combined to form error vectors $\{\mathbf{v}_i = \mathbf{z}_i - h\left(\mathbf{x}_i\right)\}$.

Pairs of such error vectors and the corresponding predictors $\boldsymbol{\phi}$ were processed using algorithm 2, and used to predict the covariances of future observations. The result of

Mean pixel value
Indicates brightness

Pixel value variance
Measures contrast

Dynamic range
Alternate measure of contrast

Pixel value entropy
Indicates structure in image

X gradient covariance
Estimates accuracy of translation registration in the x direction

Y gradient covariance
Estimates accuracy of translation registration in the y direction

R variance
Estimates accuracy of rotation registration

RS covariance
Estimates correlation between estimates of rotation/scale registration

S variance
Estimates accuracy of scale registration

X variance
Estimates accuracy of translation registration in the x direction

XY covariance
Estimates accuracy of translation registration in the y direction

Y variance
Estimates correlation between estimates of translation registration

Mean squared error between images
Indicates goodness of image alignment

Maximum absolute error between registered images
Alternate measure of goodness of image alignment

Table 5.1: *Summary of features used for covariance prediction, along with a description of their rationale*

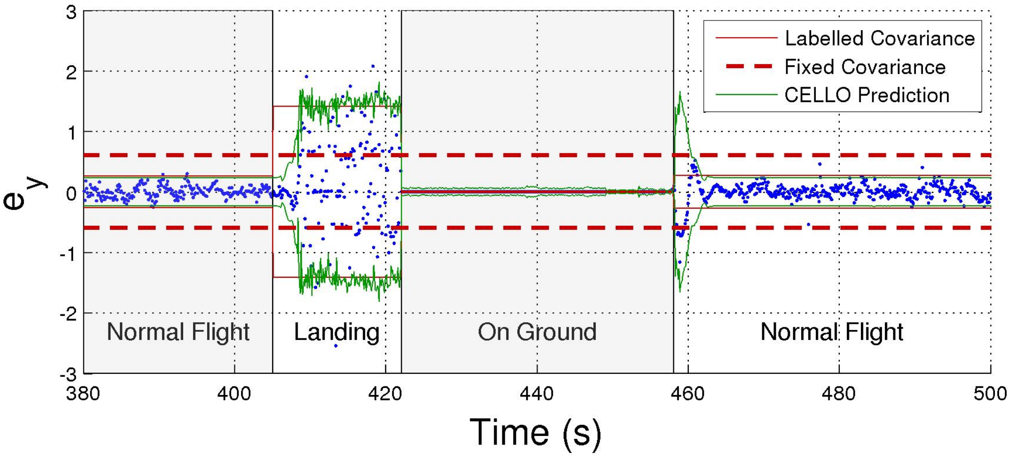


Figure 5-2: *Comparison of marginal measurement covariances for the vertical (y) component of flow. Blue dots are sample error vectors; covariances are drawn as 95% confidence margins. The labelled covariances are the empirical covariance of each region; the fixed covariance is the empirical covariance of the entire data set; the CELLO predictions are the predicted covariances for each sample. Note how the learned covariances offer increased flexibility, even within a known region; the regions of higher noise due to takeoff and landing are assigned an appropriately larger covariance.*

the learning is shown in figure 5-2. These results were attained in a few seconds on a desktop machine, given a data set of approximately ten thousand samples, corresponding to just under ten minutes of flight.

In regions where the images were of low quality, the image registration process performed erratically; this is reflected by a covariance orders of magnitude larger than during normal flight. Typically, such measurements would be rejected by heuristic outlier detectors; such heuristics require careful tuning to avoid discarding useful data while ensuring all invalid points are discarded. CELLO handles this rejection natively, assigning those points a covariance large enough to mitigate any effect they would have on a state estimate, with no tuning or other input from the user required. Note that when the vehicle is stopped and the impact of motion blur is completely eradicated, the image matching process becomes very accurate, and the predicted covariance is correspondingly small.

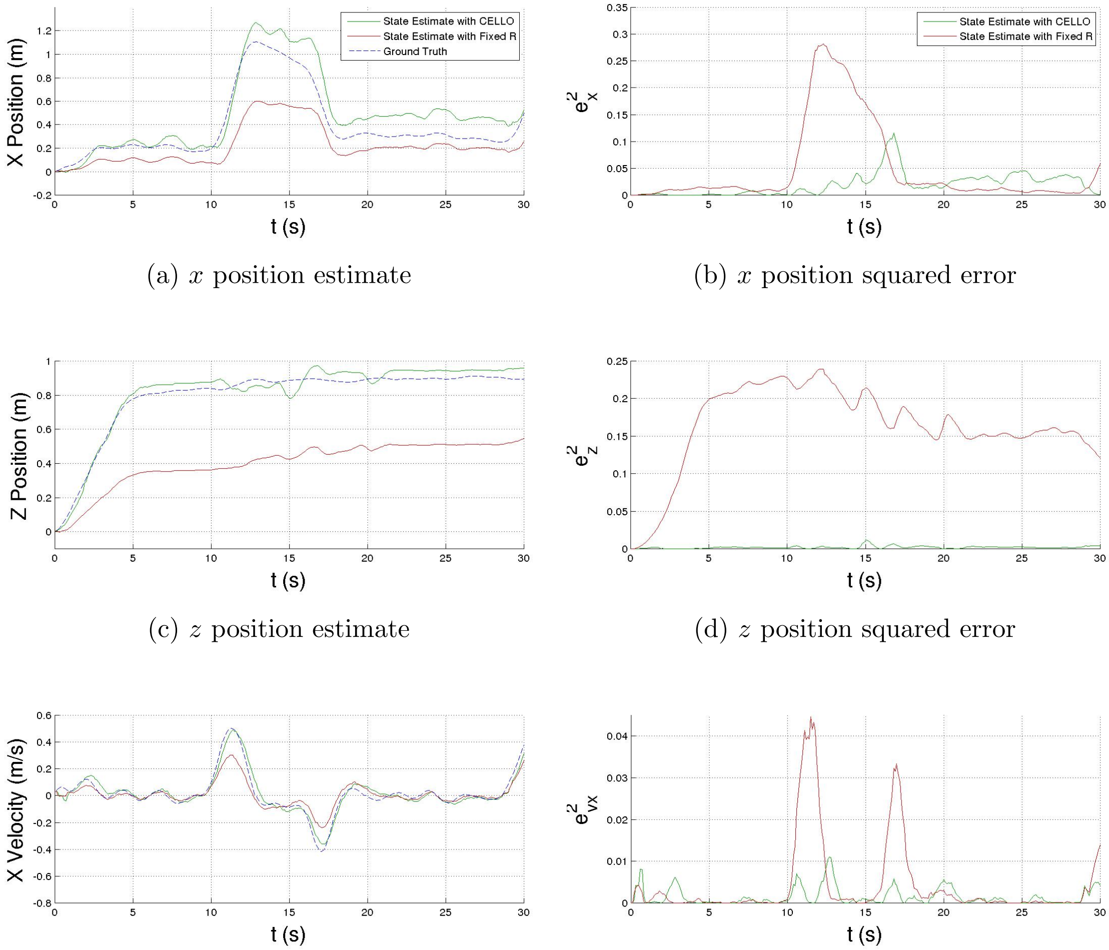
To illustrate the benefits of a predicted covariance scheme, I compared the performance of predicted and fixed covariances in online state estimation using an unscented Kalman filter. The filter incorporated both optical flow and accelerometer data. These observations are in many ways complementary; an accelerometer operates at a high frequency and with reasonable accuracy, but observes only the derivatives of the system state, and so must be integrated twice, resulting in large accumulated errors due to drift. An optical flow sensor operates at a much lower frequency, but provides measurements of velocity and height, greatly reducing drift errors. However, the covariance of the optical flow sensor is highly environment-dependent, as seen in figure 5-2. Choosing a small fixed covariance for the measurement model leads to filter divergence due to singularities in the optical flow measurement function; choosing a large covariance makes the filter slow to incorporate data. Adapting the covariance using the predictions from CELLO allows for the use of smaller covariances only when appropriate, and creates a more robust and accurate filter, as seen in figure 5-3.

5.2 Odometry in a Corridor

Recall from section 4.3 that if sequential planar laser scans are aligned, the resulting transform can be used as an indirect measurement of the vehicle velocity. However, environmental ambiguities lead such scan-matching systems to suffer from wide variations in accuracy. In a hallway-like environment, where the laser can see the walls of the hallway but not the ends, there will be low uncertainty in the direction of the walls, but high uncertainty along the length of the hallway, where information is missing. In the limiting case of parallel planar walls and no sensor noise, the transverse position in the hallway can be obtained exactly, but no information is available about the longitudinal position.

Several solutions have been developed to avoid estimator failure in hallway environments; the developers of the scan-matching algorithm I employed suggest in another paper [6] that the computed transform should be modelled as a multivariate Gaussian random variable, with a covariance chosen to match the shape of the likelihood function. This is precisely the model CELLO is designed to learn.

As in the simulated scan matching experiments done in section 4.3, I chose pre-



(a) x position estimate

(b) x position squared error

(c) z position estimate

(d) z position squared error

(e) x velocity estimate

(f) x velocity squared error

Figure 5-3: *Online state estimation performance of an unscented Kalman filter using fixed and learned covariances to integrate optical flow and accelerometer data. Only three of twelve states are displayed: the x position, the height above ground z, and the translational velocity in the x direction. Fixed covariances the empirical measurement covariances of the manually annotated sensor regime. The adapted covariances produced by CELLO allow for smaller covariances without introducing inconsistency, precluding the need for outlier rejection and allowing the filter to safely place greater confidence in new data.*

dictors consisting of two histograms. The first histogram counted the number of returned measurements in each of ten bins of angles; the second counted the angles of the line segment connecting each sequential scanned point. Together, these histograms indicated directions in which information was unavailable. The histograms were augmented with the score generated by the scan matching algorithm; a low score indicates a failed match. This additional predictor was necessary to ensure that unambiguously poor matches could be handled separately from ambiguous matches.

I used the expectation-maximization procedure described in algorithm 4 to learn a covariance model. Sample predicted covariances are drawn as 2σ ellipses in figure 5-4; note that these are representative of the observation covariance, depite being drawn in the measurement space. This presentation serves to illustrate the desired behavior; the covariance is aligned with the walls of the hallway whenever the far walls are out of range, so that observations are treated as highly uncertain in the direction information is unavailable.

Figure 5-5 compares the predictions of CELLO-EM to those of Bachrach et al.. In addition, I processed the same data using the scan matching and covariance prediction scheme developed by Andrea Censi [2], along with hand-tuned and empirical fixed covariances. Although all methods for predicting covariances agree on the direction of maximal uncertainty and on the location of the regions of high uncertainty, they vary widely in the estimated magnitude of uncertainty. To evaluate which uncertainty estimates most closely mirror the true uncertainty, I generated a proxy for ground truth by using a longer-range laser, capable of seeing the ends of the hallways traversed. These longer-ranged laser scans were processed by the SLAM algorithm presented in M. Kaess et al. [32]. This allowed the direct computation of the filter error.

I evaluated an identical filter using each covariance scheme. I additionally evaluated the performance of the filter when augmented with visual odometry observations, derived from an RGB-D camera using the method of Huang et al. [23]. Visual odometry is in many ways analogous to scan-matching odometry: feature points in successive images are aligned to compute a transform between images. This transform serves as an indirect measure of the vehicle velocity and angular velocity. For details on the methodology and implementation of this process, I refer the reader to the original paper. The ease with which I was able to extend the filter to incorporate visual odometry information showcases a key advantage of CELLO: its generality. In conjunction with the modularity of the Kalman filter, the separation of covariance prediction from state estimation makes it straightforward to improve performance by augmenting the filter with an additional sensor.

Table 5.2 compares filter performance using several metrics. The availability of an additional sensor reduced the mean error in each case, as expected. However, this came at a cost of diminished filter consistency, as measured by the normalized estimation error and the dataset likelihood. Using CELLO yields the smallest mean error, as well as the highest consistency for either metric. This clear advantage suggests that the covariances predicted by CELLO yield a superior model for the observation distribution—superior even to methods designed and tuned to predict the uncertainty of each algorithm.

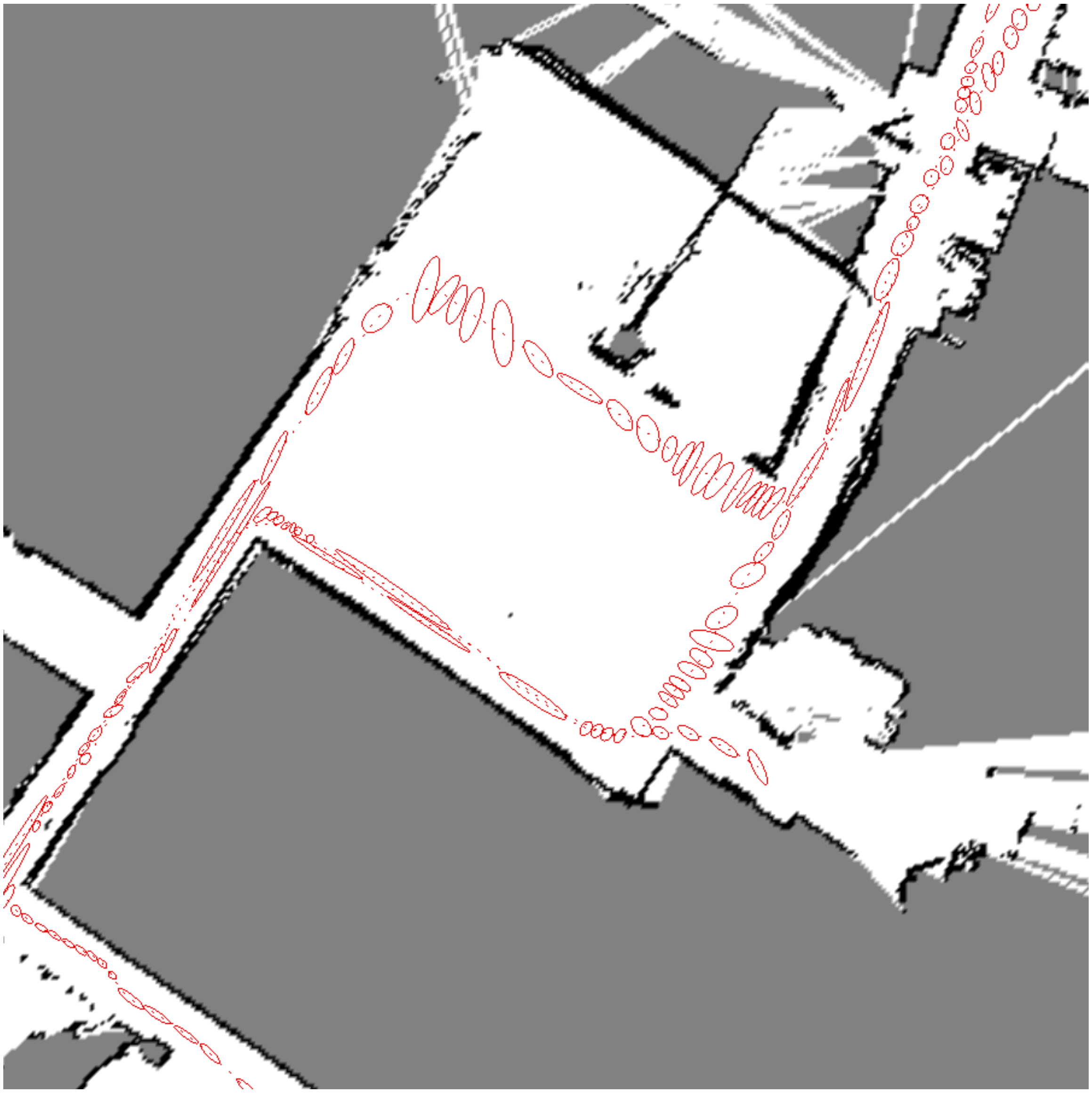


Figure 5-4: *Predicted covariances for laser scan-matching odometry in a hallway. Note that environmental ambiguity creates large uncertainty along the axis of the hallway*

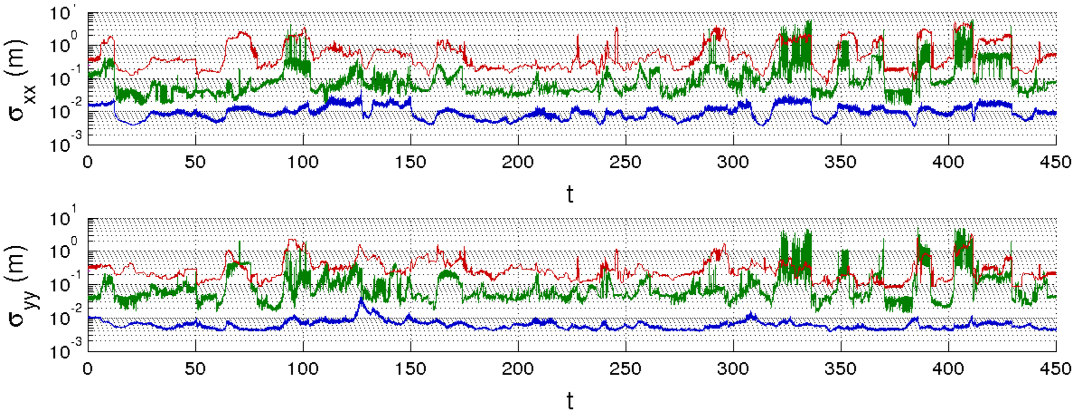


Figure 5-5: *Comparison of the marginal covariances predicted by Bachrach et al. [5] (green), Andrea Censi [2] (blue), and CELLO (red). Note the agreement across methods on the locations of regions of uncertainty, but the wide disparity in their magnitude. Using the smaller covariances results in the high variance velocity estimates seen in figure 5-6; in particular, note that the spikes in velocity variance using the covariance scheme of Bachrach coincide with regions where the covariance increases, suggesting that only CELLO sufficiently increases the covariance.*

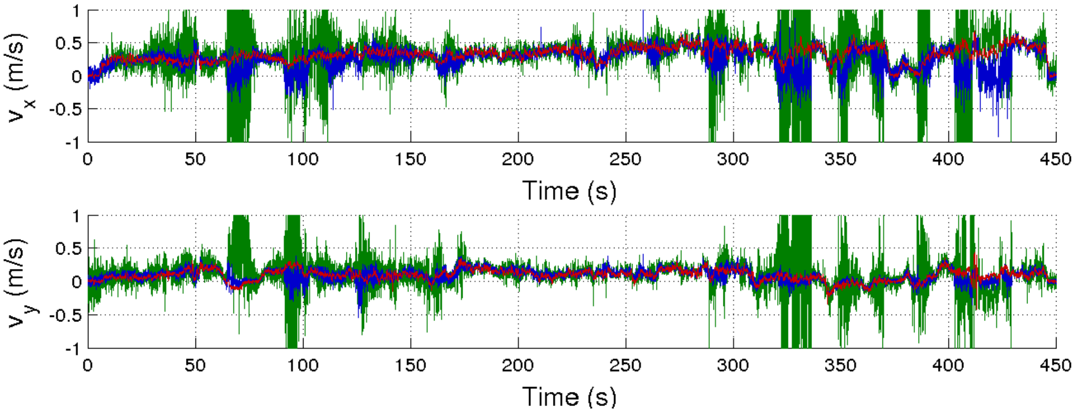


Figure 5-6: *Estimated velocity using covariances predicted by CELLO (red), Bachrach (green), Censi (blue), and fixed to hand-tuned values (purple, nearly obscured). Note the reduced variance when using CELLO.*

	RMSE[1]	MAE[2]	NEES[3]	NMEE[4]	LL[5]
Fixed	0.0303	9.1797	34.7087	1.5950	-11.0200
BSM	0.0595	18.3735	464.3963	5.1583	-356.8934
CSM	0.0407	11.9108	7025.3429	33.0318	3.1346
FOVIS	0.2397	63.0931	14431.6146	29.2197	-712.3076
BSM+FOVIS	0.0199	6.4020	1234.3538	5.3928	-620.1783
CSM+FOVIS	0.0323	9.6486	6739.2444	25.8061	-139.8690
CELLO	0.0179	6.0304	19.0661	0.9536	6.1715

Table 5.2: *Comparison of filter performance for a laser scan-matcher and optical flow system. The filter was tested using fixed covariances, using the scan matching systems of Bachrach and Censi by themselves, using the FOVIS optical flow system alone, using FOVIS in conjunction with each scan matching system, and using the scan matching system of Bachrach and FOVIS with covariances predicted by CELLO. Using CELLO results in modest gains in terms of absolute accuracy, but enormous improvements in consistency and estimator bias. All metrics are taken as specified in Bar-Shalom et al. [7]*

[1] Root mean squared error, $\sqrt{\frac{1}{K}\sum_{n=1}^{K}(\hat{\mathbf{x}}_n - \mathbf{x}_n)^\top(\hat{\mathbf{x}}_n - \mathbf{x}_n)}$. Low values indicate an accurate estimator; lower is better.

[2] Mean absolute error, $\frac{1}{K}\sum_{n=1}^{K}\sum_{i=1}^{N_x}|(\hat{x}_i - x_i)_n|$. Low values indicate an accurate estimator; lower is better.

[3] Normalized estimation error squared, $\frac{1}{KN_x}\sum_{n=1}^{K}(\hat{\mathbf{x}}_n - \mathbf{x}_n)^\top\mathbf{\Sigma}_n^{-1}(\hat{\mathbf{x}}_n - \mathbf{x}_n)$. A lower value indicates a more consistent estimator.

[4] Normalized mean estimation error, $\frac{1}{K}\sum_{i=1}^{N_x}\left|\sum_{n=1}^{K}\frac{(\mathbf{x}_i)_n}{(\mathbf{\Sigma}_{ii})_n}\right|$. A value of zero implies an unbiased estimator for all states; lower is better.

[5] Normalized log likelihood, $\frac{1}{K}\sum_{i=1}^{K}\log p(\mathbf{z}_i|\mathbf{R}_i)$. High values indicate a good measurement model; higher is better.

Chapter 6

Conclusion

I have presented a solution to a long-standing problem in recursive parametric estimation. Sensor measurements are generally coupled to both the environment and to the vehicle state. By reasoning about that coupling in an abstract way, I develop a general algorithmic solution to the problem of identifying the noise parameters of a time-varying model, while avoiding the need to infer a distribution over the environment.

The principle theoretical contribution of this thesis is the notion that introducing auxiliary information, information which would have otherwise been discarded, can improve the fidelity of graphical models in a very general way, and can have a significant impact on real-world performance. I present an algorithm for tractable inference on the models induced by the introduction of an auxiliary predictor vector $\boldsymbol{\phi}$. This algorithm has four principle strengths. First, it is computationally cheap. At run-time, the cost of a parameter prediction is essentially the cost of a nearest-neighbor search: $\mathcal{O}(\log N_D)$. Even under a Bayesian interpretation, we do not require sampling or other expensive procedures. A modern computer can handle searches like this in real time even for data sets containing hundreds of thousands, or even millions, of samples. In addition, carefully discarding samples that do not supply new information allows the system to achieve dense coverage of the predictor space without an excessively large data set.

Second, the algorithm may be easily adapted to new sensors. The entire process required to use CELLO on a new sensor has just three steps.

1. Choose a parametric model for the sensor
2. Choose a vector of predictor features, computable from available sensor date
3. Collect a body of data and feed it to the CELLO learning subroutine

The predictor vector need not be minimal; a large set of parameters can be provided to the learning subroutine, and any features which are insufficiently informative will be automatically discarded. The only experimental step is the collection of routine operating data; the computational step, assuming the designer uses an existing implementation of CELLO, is just a single command. This is much simpler than difficulty involved in generating from first principles a covariance model for a new sensor.

Third, the algorithm is noninvasive, in the sense that it augments, rather than disrupts, existing filtering algorithms. An implementation of the Kalman filter can be adapted to use CELLO in just a few lines of code, simply replacing the call to a static covariance matrix with a call to the CELLO prediction subroutine. This represents very little coder effort; the only remaining cost to the designer is the need to collect data and run the CELLO training subroutine, which only need be done once. Decisions about optimizations done within the filter may be made independently of CELLO; for instance, the Kalman filter could be replaced with an extended, unscented, or cubature Kalman filter and use exactly the same calls to CELLO.

Fourth, if run on-line, the algorithm is globally asymptotically correct. That is, given sufficient samples, the predictions will be consistent; the bias and variance of the predicted parameters will collapse to zero. In contrast, parameter estimation systems derived from a detailed model of the target sensor may perform arbitrarily poorly if the model is inaccurate.

The algorithm is not without limitation. First, the method asserts a parametric form for the observation distribution which may or may not reflect reality. We may prove consistency if the form is correct, but if the observation noise is non-Gaussian—if, for instance, it is heavy-tailed, or multi-modal, or heavily skewed—the algorithm may perform arbitrarily poorly. This is true for any parametric solution, of course; the only way to avoid this pitfall is to use nonparametric models. In general, nonparametric models are more computationally expensive than recursive parametric algorithms; however, they can make stronger performance guarantees.

Second, by its nature the algorithm is excessively general. The user must provide a space of predictors, a metric on the predictor space, and a scalar kernel function which operates on that metric. It is difficult to provide principled guidance for how these should be chosen. The learning procedure allows for the user to provide a parametric form for the metric and kernel; in many cases, simple parametric forms like the scaled Euclidean metric and the linear kernel perform well enough that more complex forms need not be considered. If they fail to perform well, the procedure for choosing a more complex form is not clear.

Ideally, the learning procedure would be extended to include learning predictor features as well. In my work, I simply made educated guesses as to which features would be informative; for many sensors, choosing an excessively large set and allowing the metric learning process to discard uninformative features worked well. However, I had domain knowledge to draw upon; I knew which failure modes I was attempting to identify, and what features could identify those modes. Given an arbitrary sensor with a known observation function, it is not immediately apparent what good features might be. Feature discovery is an active area of research in machine learning; augmenting CELLO with a system for feature discovery would extend its the applicability.

Finally, the algorithm relies on nearest neighbor searches, which often perform poorly in high-dimensional spaces. If the predictor feature dimensionality exceeds ten or twelve, fast searches, like the k-D tree I use, will perform no better than a naïve linear search. If the algorithm is run in online mode, the dataset will grow without bound, and this linear search can become prohibitively expensive; this problem is common to many applications which rely on fast search, such as the rapidly-exploring random tree algorithms used in planning. Although this cost may be mitigated by only adding the most informative features, it remains significant, and dominates the computational burden of the algorithm. Development of improved heuristics for structuring the dataset, which increase the number of dimensions we may effectively search, could dramatically improve the performance of the algorithm. Alternatively, it may be possible to extract exemplars and dramatically decrease the size of the dataset, without compromising performance; this would enable the algorithm to be employed even on systems with very limited memory, such as microcontrollers.

To my knowledge, no other solution can claim the same mix of correctness, ease of use, and tractability. I believe I have presented a method which can be used as an 'off-the-shelf' tool for improving estimation with minimal overhead, either on the part of the coder or on the part of the computer. The ability to bootstrap an estimator with no tuning can dramatically reduce the effort required to get a system up and running, while augmenting—rather than penalizing—the performance. The flexibility of the observation model and the generality of my solution may be useful in a variety of contexts beyond state estimation; the algorithm derived promises to improve accuracy and consistency anywhere a Gaussian measurement model is used to account for the uncertainty of a complex system.

Appendix A

Software Implementation

In order to evaluate CELLO and CELLO-EM, I implemented both algorithms in C++. Several practical design and usage considerations emerged during the development process. I record these concerns and how they were addressed, to guide anyone who may implement CELLO, or use my implementation, in the future.

A.1 Ensuring positive-definiteness

Because CELLO fundamentally represents a covariance as a (weighted) sum of outer products, the resulting prediction is guaranteed to be positive semidefinite. In particular, for an observation of size $N_{\mathbf{z}}$, the prediction will be almost surely positive definite if more than $N_{\mathbf{z}}$ samples are incorporated into the estimate, and will always be semidefinite if the number of samples is less than $N_{\mathbf{z}}$. This poses a problem: for any valid distribution over the positive definite cone, the set of semidefinite matrices has measure zero. For kernels with infinite support, this is an issue only if numerical precision comes into play. However, for finite support kernels and finite data sets, our system will, in some regions of predictor space, with probability one generate an estimate with probability zero.

This is a practical problem, more than a theoretical one. In many cases, we require the inverse of the covariance matrix, which is undefined for semidefinite matrices. There is always a valid, simple solution: collect more data. However, in practice this may not be feasible, especially if some region of the space of predictors has very low, but nonzero, probability. One simple solution is to redefine our estimator (equation (3.22)) as

$$\hat{\mathbf{R}}(\boldsymbol{\phi}, \boldsymbol{\alpha}) = \frac{\epsilon\mathbb{1} + \sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|)\boldsymbol{\alpha}\mathbf{v}\mathbf{v}^\top}{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|)\boldsymbol{\alpha}} \tag{A.1}$$

where ϵ is a very small number. In the limit of infinite samples, this exactly recovers the CELLO prediction. [1]

The practical implication of this modification is to ensure a positive definite prediction in all cases. In particular, it ensures the likelihood will always be finite; a semidefinite covariance matrix will yield a sample likelihood of $-\infty$ if the projection of the sample onto the null space of the covariance is not zero.

A.2 Guiding the search

Because the search space is large and non-convex, we can improve the optimization process by adding additional terms to the cost function, with the intention of guiding the search. The software implementation referred to does this in three ways.

First, for very large scale parameters, the kernel sum $\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|)\boldsymbol{\alpha}$ will become very small for large segments of the predictor space. This can cause the Jacobian (equation (3.36)) to become very small, leading the optimizer to 'get lost'. Since numerical optimizers often use the magnitude of the Jacobian as a stopping condition, this can lead to the optimizer returning hyperparameters which perform extremely poorly. By adding an L2 regularization term to the cost function, we can guide the optimizer away from these false solutions.

$$\boldsymbol{\alpha}^\star = \arg\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) + k_2\|\boldsymbol{\alpha}\|_2 \tag{A.2}$$

Second, we observe that the estimator variance decreases in regions of high density

[1] Actually, ϵ may be any number, and in the limit of infinite samples we will recover our original estimator. However, for finite sample sets, small ϵ will have less of an effect in low-probability regions of predictor space.

sample density. We may capitalize on this fact by explicitly rewarding the mean sample density with reduced cost. This will tend to guide the hyperparameters in the direction which most increases sample density without overly decreasing likelihood. By implication, this sample density reward serves as a form of regularization, and will help to combat overfitting.

$$\boldsymbol{\alpha}^{\star} = \arg\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) + k_2\|\boldsymbol{\alpha}\|_2 + k_{\sum} \sum_j \log\left(\sum_i k(\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|)\boldsymbol{\alpha}\right) \tag{A.3}$$

Note that the gradient of this particular form of a low density cost term also appears in the gradient of the likelihood; we achieve this additional regularization with zero computational overhead.

Finally, it is not always readily apparent which features will be informative. The form of the estimator allows the rejection of uninformative features by simply making the corresponding hyperparameter weight arbitrarily small. If that weight is not just small, but zero, we may safely discard the feature. Because nearest neighbor searches have much higher performance in lower dimensional spaces, the ability to discard features can lead to large speedups. To encourage sparse feature weights, we add an L1 regularization term to the cost function. This will have minimal effect for large feature weights, but will encourage small weights to become exactly zero.

$$\boldsymbol{\alpha}^{\star} = \arg\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) + k_1\|\boldsymbol{\alpha}\|_1 + k_2\|\boldsymbol{\alpha}\|_2 + k_{\sum} \sum_j \log\left(\sum_i k(\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|)\boldsymbol{\alpha}\right) \tag{A.4}$$

The cost function gains $\{k_1, k_2, k_{\sum}\}$ must be tuned during optimization to achieve good performance. We found the values of $k_1 = 10^{-5}$, $k_2 = 10^{-3}$, and $k_{\sum} = 10^{-}2$ good general purpose initial values; as a rule of thumb, k_2 should be made as small as possible, and increased only if the optimization fails. k_1 and $k_{\sum}$ reflect performance considerations—the preference for low dimension and high density, respectively—and should be tuned accordingly.

A.3 Optimizations

In order to deal with very large datasets in real time, it is important to be able to quickly evaluate the sums in our estimator. Using kernels with finite support causes many terms in the sum to become zero. Running a fixed-radius nearest neighbor search allows the rapid identification of the nonzero terms. If the hyperparameters are fixed, or fixed up to an overall scale, this may be done on $\mathcal{O}(N_{\text{neighbors}} \log N_D)$ time using a vantage point tree for arbitrary metrics. If the metric is a scaled Euclidean, this may be accelerated further by using a k-D tree. Note that a naive implementation would take $\mathcal{O}(N_D)$ time, regardless of the number of neighbors $N_{\text{neighbors}}$. If the dimension is high, approximate nearest neighbor searches often dramatically outperform exact nearest neighbor searches. I used the library FLANN [35], which provides tools for both approximate and exact nearest neighbor search under arbitrary metrics and automatically tunes for speed.

A.4 Parallelization

CELLO, and the maximization step of CELLO-EM, can be easily parallelized in two ways. First, since the optimization is non-convex, there may be multiple maxima. Gradient ascent and related methods will thus be sensitive to initial conditions. Using random restarts improves the chance of finding the global maxima; each restart is independent from the others, and so they may be done in parallel, with the maximum likelihood hyperparameters $\boldsymbol{\alpha}^{\star}$ taken as the supremum of the set of local maxima.

Additionally, the algorithm can take advantage of low-level parallelism. The evaluation of the objective function, or its gradient, requires the summation of a sequence of independent terms. Each term requires access to the entire data set, but each term may be evaluated independently. Because these terms are independent, we may arbitrarily partition the dataset, and evaluate the objective function for each partition, then sum the result. Note that the L1 and L2 terms must be evaluated separately, if present. However, this is a negligible computational cost compared to computing the summed terms.

Both forms of parallelization require minimal interprocess communication; every thread needs access to the full dataset, but needs no further communication beyond that. I found that evaluating the sums serially but running optimizations in parallel resulted in the greatest absolute increase in speed; however, my experiments were done on a four core desktop machine. Conceivably, a computer with many cores—such as a cluster or a GPU—could achieve a greater speed increase using both forms of parallelization.

Appendix B

Derivations

B.1 Proof of convergence of kernel density estimates

Let $\mathcal{D}$ be a set of N_D independent, identically distributed random variables $\{\mathbf{x}_1, \ldots, \mathbf{x}_{N_D}\}$, taking values in a measureable space $\mathcal{X}$. Let the density of each $\mathbf{x}_i$ be $p(\mathbf{x})$. Then the *kernel density estimate* of the distribution $p(\mathbf{x})$ given the data set $\mathcal{D}$ and a *kernel* $k(\cdot, \cdot)$ is defined in equation (B.1).

$$\hat{p}(\mathbf{x}) = \frac{1}{N_D} \sum_{i=1}^{N_D} k(\mathbf{x}, \mathbf{x}_i). \tag{B.1}$$

A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is an admissible kernel if it has the following properties.

Non-negative:	$k(\mathbf{x}, \mathbf{x}') \geq 0$	$\forall \mathbf{x}, \mathbf{x}'$
Normalized:	$\int_{\mathcal{X}} \mathrm{d}\mathbf{x}' k(\mathbf{x}, \mathbf{x}') = 1$	$\forall \mathbf{x}$
Asymptotically identical:	$\lim_{N_D \to \infty} k(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x} - \mathbf{x}')$	$\forall \mathbf{x}, \mathbf{x}'$
Exponentially bounded:	$\lim_{\Vert\mathbf{x} - \mathbf{x}'\Vert \to \infty} \exp(\lambda \Vert\mathbf{x} - \mathbf{x}'\Vert) k(\mathbf{x}, \mathbf{x}') < \infty$	$\exists \lambda > 0, \forall \mathbf{x}, \mathbf{x}'$

Here, the delta function $\delta(\cdot)$ is the Dirac measure, equal to the Dirac delta function if $\mathcal{X}$ is a Euclidean space. Note that it follows from these conditions that if $k_1(\mathbf{x}_1, \mathbf{x}'_1)$ is a kernel on $\mathcal{X}_1$ and $k_2(\mathbf{x}_2, \mathbf{x}'_2)$ is a kernel on $\mathcal{X}_2$, the product $k_1(\mathbf{x}_1, \mathbf{x}'_1)k_2(\mathbf{x}_2, \mathbf{x}'_2)$ is a kernel on the product space $\mathcal{X}_1 \times \mathcal{X}_2$. The first two condtions guarantee that the kernel density estimate will be a valid distribution for any nonzero number of samples; I will now show that the last two properties guarantee the estimate will be consistent.

The mean and variance of the kernel density estimate follow from equation (B.1).

$$\begin{aligned}\mathbf{E}\left[\hat{p}(\mathbf{x})\right] &= \int \prod_{j=1}^{N_D} \mathrm{d}\mathbf{x}_j p(\mathbf{x}_j) \, \frac{1}{N_D} \sum_{i=1}^{N_D} k(\mathbf{x}, \mathbf{x}_i) \\ &= \frac{1}{N_D} \sum_{i=1}^{N_D} \int \mathrm{d}\mathbf{x}_i p(\mathbf{x}_i) \, k(\mathbf{x}, \mathbf{x}_i) \\ &= \int \mathrm{d}\mathbf{x}' p(\mathbf{x}') \, k(\mathbf{x}, \mathbf{x}') \end{aligned} \tag{B.2}$$

Therefore, the kernel density estimate is unbiased if and only if the kernel function $k(\mathbf{x}, \mathbf{x}')$ approaches the delta function $\delta(\mathbf{x}' - \mathbf{x})$ as the number of samples increases.

$$\forall k(\cdot,\cdot) : \lim_{N_D \to \infty} k(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x}' - \mathbf{x}), \qquad \lim_{N_D \to \infty} \mathbf{E}\left[\hat{p}(\mathbf{x})\right] = p(\mathbf{x}) \tag{B.3}$$

Taking the variance and simplifying:

$$\begin{aligned}\mathbf{V}\left(\hat{p}(\mathbf{x})\right) =& \int \prod_{l=1}^{N_D} \mathrm{d}\mathbf{x}_l p(\mathbf{x}_l) \left(\frac{1}{N_D} \sum_{i=1}^{N_D} k(\mathbf{x}, \mathbf{x}_i) \right)^2 - \mathbf{E}\left[\hat{p}(\mathbf{x})\right]^2 \\ =& \frac{1}{N_D^2} \left(\sum_i \int \mathrm{d}\mathbf{x}_i p(\mathbf{x}_i) \, k(\mathbf{x}, \mathbf{x}_i)^2 + 2 \sum_{i \neq j} \left(\int \mathrm{d}\mathbf{x}_i p(\mathbf{x}_i) \, k(\mathbf{x}, \mathbf{x}_i) \right)^2 \right) - \mathbf{E}\left[\hat{p}(\mathbf{x})\right]^2 \\ =& \frac{1}{N_D} \left(\int \mathrm{d}\mathbf{x}' p(\mathbf{x}') \, k(\mathbf{x}, \mathbf{x}')^2 - \left(\int \mathrm{d}\mathbf{x}' p(\mathbf{x}') \, k(\mathbf{x}, \mathbf{x}') \right)^2 \right) \end{aligned} \tag{B.4}$$

Because the kernel is exponentially bounded, the parenthetical term in equation (B.4) is guaranteed to be finite; denoting it $f(\mathbf{x})$,

$$f(\mathbf{x}) = \left(\int \mathrm{d}\mathbf{x}' p(\mathbf{x}')\, k(\mathbf{x}, \mathbf{x}')^2 - \left(\int \mathrm{d}\mathbf{x}' p(\mathbf{x}')\, k(\mathbf{x}, \mathbf{x}') \right)^2 \right) \tag{B.5}$$

it is apparent the variance must become zero as the number of samples goes to infinity.

$$\lim_{N_D \to \infty} \mathbf{V}\left(\hat{p}(\mathbf{x})\right) = \lim_{N_D \to \infty} \frac{f(\mathbf{x})}{N_D} = 0 \tag{B.6}$$

Therefore, if a kernel is exponentially bounded and approaches a delta function, the kernel density estimate is consistent.

B.2 Derivation of asymptotic estimator properties

Let the vector $\boldsymbol{\theta}(\boldsymbol{\phi}) \in \mathbb{R}^{\frac{N_{\mathbf{z}}(N_{\mathbf{z}}+1)}{2}}$ represent the independent elements of the observation covariance matrix $\mathbf{R} = R(\boldsymbol{\phi})$ associated with a predictor vector $\boldsymbol{\phi}$.

$$\boldsymbol{\theta}(\boldsymbol{\phi}) = \mathrm{vec}\,\left[R(\boldsymbol{\phi})\right] \tag{B.7}$$

Additionally, let the function $\mathbf{T}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ represent the corresponding elements of the matrix estimator $\mathbf{T}_{\mathbf{R}}(\mathbf{x}, \mathbf{z}) = (\mathbf{z} - H\mathbf{x})(\mathbf{z} - H\mathbf{x})^\top$ so that

$$\mathbf{T}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = \mathrm{vec}\,\left[\mathbf{T}_{\mathbf{R}}(\mathbf{x}, \mathbf{z})\right] \tag{B.8}$$

It follows from equation (3.6) that if $\mathbf{z} \sim \mathcal{N}\left(\mathbf{H}\mathbf{x}, R(\boldsymbol{\phi})\right)$—that is, if the tuple $\{\mathbf{x}, \mathbf{z}, \boldsymbol{\phi}\}$ is a sample from a CELLO dataset, as introduced in chapter 3—then $\mathbf{T}_{\boldsymbol{\theta}}$ is an unbiased estimator of the parameter vector $\boldsymbol{\theta}$.

$$\mathbf{E}\left[\mathbf{T}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] = \boldsymbol{\theta}(\boldsymbol{\phi}) \tag{B.9}$$

Using this notation, the estimator presented in algorithm 1 can be written

$$\hat{\boldsymbol{\theta}}(\boldsymbol{\phi}) = \frac{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|)\mathbf{T}_{\boldsymbol{\theta}}(\mathbf{v}_i)}{\sum_{i=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|)} \tag{B.10}$$

The expected value of this estimator, given N_D sample triplets, is given explicitly in equation (B.11).

$$.\mathbf{E}\left[\hat{\boldsymbol{\theta}}(\boldsymbol{\phi})\right] = \int \prod_{i=1}^{N_D} p(\mathbf{x}_i)\, p(\boldsymbol{\phi}_i)\, p(\mathbf{z}_i|\boldsymbol{\phi}_i, \mathbf{x}_i)\, \mathrm{d}\mathbf{x}_i \mathrm{d}\mathbf{z}_i \mathrm{d}\boldsymbol{\phi}_i \frac{\sum_{j=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_j\|)\mathbf{T}_{\boldsymbol{\theta}}(\mathbf{x}_j, \mathbf{z}_j)}{\sum_{l=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_l\|)} \tag{B.11}$$

$$= \int \prod_{i=1}^{N_D} p(\boldsymbol{\phi}_i)\, \mathrm{d}\boldsymbol{\phi}_i \frac{\sum_{j=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_j\|)\boldsymbol{\theta}(\boldsymbol{\phi}_j)}{\sum_{l=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_l\|)} \tag{B.12}$$

$$= \sum_{j=1}^{N_D} \int \mathrm{d}\boldsymbol{\phi}_j\, p\big(\boldsymbol{\phi}_j\big)\, k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_j\|)\boldsymbol{\theta}(\boldsymbol{\phi}_j) \int \prod_{i \neq j} \frac{p(\boldsymbol{\phi}_i)\, \mathrm{d}\boldsymbol{\phi}_i}{\sum_{l=1}^{N_D} k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_l\|)} \tag{B.13}$$

The simplified form of equation (B.12) follows from equation (B.9): since $\mathbf{T}_{\boldsymbol{\theta}}$ is an unbiased estimate of $\boldsymbol{\theta}$, the expectation over the state $\mathbf{x}$ and the observations $\mathbf{z}$ can be immediately simplified. Equation (B.13) follows from the independence of the samples and properties of the integral.

Note that this integral is the standard mean function integral of a Nadaraya-Watson estimator; at this stage, the stochastic nature of the regression target, $\boldsymbol{\theta}$, is unimportant. Despite the fact that we cannot directly observe $\boldsymbol{\theta}$, it is possible to apply standard regression techniques and results. I provide a derivation of the bias, and just the result of the variance; a more complete derivation can be found in any text on kernel regression.

Denote the kernel scale as σ and assume the kernel metric to be of *generalized Euclidean form*, with a metric tensor $\mathbf{M}$:

$$\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\| = \sqrt{(\boldsymbol{\phi} - \boldsymbol{\phi}_i)^\top \mathbf{M} (\boldsymbol{\phi} - \boldsymbol{\phi}_i)} \tag{B.14}$$

This permits the definition of a local coordinate system $\boldsymbol{\varphi} = \frac{1}{\sigma}\mathbf{L}(\boldsymbol{\phi} - \boldsymbol{\phi}_i)$, where $\mathbf{M} =$

$\mathbf{L}^\top\mathbf{L}$; in this local coordinate system, the kernel function is spherically symmetric.

$$k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|_{\mathbf{M}}, \sigma) = \sigma k(\|\boldsymbol{\varphi}\|) \tag{B.15}$$

Taking the Taylor series about $\boldsymbol{\phi}$, the k^{th} parameter of the parameter vector $\boldsymbol{\theta}$, θ_k can be approximated as

$$\theta_k(\boldsymbol{\phi}_i) = \theta_k(\boldsymbol{\phi}) + \nabla^\top\theta_k(\boldsymbol{\phi})[\boldsymbol{\phi} - \boldsymbol{\phi}_i] + \frac{1}{2}[\boldsymbol{\phi} - \boldsymbol{\phi}_i]^\top\nabla\nabla^\top\theta_k(\boldsymbol{\phi})[\boldsymbol{\phi} - \boldsymbol{\phi}_i] + \mathcal{O}\left(\|\boldsymbol{\phi} - \boldsymbol{\phi}_i\|^3\right) \tag{B.16}$$

$$= \theta_k(\boldsymbol{\phi}) + \sigma\nabla^\top\theta_k(\boldsymbol{\phi})\mathbf{L}^{-1}\boldsymbol{\varphi}_i + \frac{\sigma^2}{2}\boldsymbol{\varphi}_i^\top\mathbf{L}^{-1}\nabla\nabla^\top\theta_k(\boldsymbol{\phi})\mathbf{L}^{-1}\boldsymbol{\varphi}_i + \mathcal{O}\left(\sigma^3\|\boldsymbol{\varphi}_i\|^3\right) \tag{B.17}$$

where $\nabla^\top$ is the gradient operator and $\nabla\nabla^\top$ the Hessian. The predictor distribution can likewise be expressed as a Taylor series.

$$p(\boldsymbol{\phi}_i) = p(\boldsymbol{\phi}) + \sigma\nabla^\top p(\boldsymbol{\phi})\mathbf{L}^{-1}\boldsymbol{\varphi}_i + \frac{\sigma^2}{2}\boldsymbol{\varphi}_i^\top\mathbf{L}^{-1}\nabla\nabla^\top p(\boldsymbol{\phi})\mathbf{L}^{-1}\boldsymbol{\varphi}_i + \mathcal{O}\left(\sigma^3\|\boldsymbol{\varphi}_i\|^3\right) \tag{B.18}$$

Because the kernel is exponentially bounded, it follows that the higher order terms are finite when integrated with the kernel.

$$\int \mathrm{d}\boldsymbol{\varphi} k(\|\boldsymbol{\varphi}\|)\mathcal{O}\left(\|\boldsymbol{\varphi}\|^3\right) < \infty \tag{B.19}$$

Substituting these expansions into equation (B.13), changing variables to the local coordinate system $\boldsymbol{\varphi}$, and rearranging to expose higher order terms,

$$\begin{aligned}\mathbf{E}\left[\hat{\theta}_k(\boldsymbol{\phi})\right] = \sum_{j=1}^{N_D}\int \mathrm{d}\boldsymbol{\varphi}\Bigg(& p\theta_k + \sigma\Big(\theta_k\nabla^\top p + p\nabla^\top\theta_k\Big)\mathbf{L}^{-1}\boldsymbol{\varphi}_j \\ &+ \frac{\sigma^2}{2}\boldsymbol{\varphi}_j^\top\mathbf{L}^{-1}\Big(\theta_k\nabla\nabla^\top p + \nabla\theta_k\nabla^\top p + p\nabla\nabla^\top\theta_k\Big)\mathbf{L}^{-1}\boldsymbol{\varphi}_j \\ &+ \mathcal{O}\left(\sigma^3\|\boldsymbol{\varphi}_j\|^3\right)\Bigg)k(\|\boldsymbol{\varphi}\|)\int\prod_{i\neq j}\frac{p(\boldsymbol{\phi}_i)\,\mathrm{d}\boldsymbol{\phi}_i}{\sum_{l=1}^{N_D}k(\|\boldsymbol{\phi} - \boldsymbol{\phi}_l\|)}\end{aligned} \tag{B.20}$$

Because the kernel is isotropic, all first order terms vanish by symmetry. Define

$\nabla^2_{\mathbf{M}}\boldsymbol{\theta}_i(\boldsymbol{\phi}) = \operatorname{tr}\left[\nabla\nabla^\top\boldsymbol{\theta}_i(\boldsymbol{\phi})\mathbf{M}^{-1}\right]$ and $c_k = \int \varphi_i^2 k(\boldsymbol{\varphi})\mathrm{d}\boldsymbol{\varphi}$ for convenience.

$$\mathbf{E}\left[\hat{\theta}_k(\boldsymbol{\phi})\right] = \sum_{j=1}^{N_D}\left(p\theta_k + \frac{\sigma^2 c_k}{2}\left(\theta_k\nabla^2_{\mathbf{M}}p + 2\nabla\theta_k\mathbf{M}^{-1}\nabla^\top p + p\nabla^2_{\mathbf{M}}\theta_k\right)\right)$$
$$\int\prod_{i\neq j}\frac{p(\boldsymbol{\phi}_i)\,\mathrm{d}\boldsymbol{\phi}_i}{\sum_{l=1}^{N_D}k(\|\boldsymbol{\phi}-\boldsymbol{\phi}_l\|)} + \mathcal{O}\left(\sigma^3\right) \tag{B.21}$$

The denomenator in the fractional term is proportional to the kernel density estimate, $\hat{p}(\boldsymbol{\phi})$. Rewriting the expectation in terms of the inverse expectation of the kernel density estimate:

$$\mathbf{E}\left[\hat{\theta}_k(\boldsymbol{\phi})\right] = \mathbf{E}\left[\frac{p(\boldsymbol{\phi})}{\hat{p}(\boldsymbol{\phi})}\right]\left(\theta_k + \frac{\sigma^2 c_k}{2}\left(\frac{\theta_k}{p}\nabla^2_{\mathbf{M}}p + \frac{2}{p}\nabla\theta_k\mathbf{M}^{-1}\nabla^\top p + \nabla^2_{\mathbf{M}}\theta_k\right)\right) + \mathcal{O}\left(\sigma^3\right) \tag{B.22}$$

The consistency of the kernel density estimate implies that for large samples sizes the lead term is one. Therefore, in the limit of many samples and small scale, the estimator is unbiased to first order.

$$\lim_{\substack{\sigma\to 0\\ N\sigma\to\infty}}\mathbf{E}\left[\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}(\boldsymbol{\phi})\right] = \left(\nabla^2_{\mathbf{M}}\boldsymbol{\theta}(\boldsymbol{\phi}) + 2\nabla\log p(\boldsymbol{\phi})^\top\,\mathbf{M}^{-1}\nabla\boldsymbol{\theta}(\boldsymbol{\phi}) + \nabla^2_{\mathbf{M}}p(\boldsymbol{\phi})\,\boldsymbol{\theta}(\boldsymbol{\phi})\right)\frac{\sigma^2 c_K}{2} \tag{B.23}$$

The procedure for evaluating the asymptotic variance is similar, and a full treatment can be found in any standard text on kernel regression; the result is given in equation (B.24).

$$\lim_{\substack{\sigma\to 0\\ N\sigma\to\infty}}\mathbf{V}\left(\hat{\boldsymbol{\theta}}\right) = \frac{d_K}{p(\boldsymbol{\phi})\,N_D\sigma}\mathbf{V}\left(\boldsymbol{\theta}_i|\boldsymbol{\phi}\right) \tag{B.24}$$

Here, $d_k = \int k(\boldsymbol{\varphi})^2\mathrm{d}\boldsymbol{\varphi}$. In the limit of large samples size and small kernel scale, both the bias and variance become zero; we have obtained a consistent estimator for the covariance $\mathbf{R}$.

B.3 Derivation of Bayesian measurement model

We aim to choose the maximum entropy distribution subject to a constraint on the Kullback-Liebler divergence. That is, we seek

$$p(\mathbf{z}|\mathbf{R}', \boldsymbol{\phi}, \boldsymbol{\phi}') = \arg\max_p \mathcal{H}[p] = \arg\max_p \int \mathrm{d}\mathbf{z} - p(\mathbf{z}) \log p(\mathbf{z}) \tag{B.25}$$

subject to

$$0 = \int \mathrm{d}\mathbf{z}\, p(\mathbf{z}|\mathbf{R}', \boldsymbol{\phi}, \boldsymbol{\phi}') \left(D_{\mathrm{KL}} \left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \right) - \log \left(\frac{p(\mathbf{z}|\mathbf{R}', \boldsymbol{\phi}, \boldsymbol{\phi}')}{\mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)} \right) \right) \tag{B.26}$$

in additon to a normalization condition.

$$1 = \int \mathrm{d}\mathbf{z} p(\mathbf{z}|\mathbf{R}', \boldsymbol{\phi}, \boldsymbol{\phi}') \tag{B.27}$$

We use the method of Lagrange multipliers to transform this to an unconstrained optimization.

$$p(\mathbf{z}|\mathbf{R}', \boldsymbol{\phi}, \boldsymbol{\phi}') = \arg\max_p \mathcal{L}[p](\mathbf{z}) \tag{B.28}$$

$$\mathcal{L}[p](\mathbf{z}) = \int \mathrm{d}\mathbf{z} p(\mathbf{z}) \log p(\mathbf{z}) + \lambda_{KL} p(\mathbf{z}) \left(D_{\mathrm{KL}} \left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \right) - \log \left(\frac{p(\mathbf{z})}{\mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)} \right) \right) + \lambda_2 p(\mathbf{z}) \tag{B.29}$$

$$= \int \mathrm{d}\mathbf{z} \; - (1 + \lambda_{KL})(p(\mathbf{z}) \log p(\mathbf{z})) + p(\mathbf{z}) \left(\lambda_{KL} \left(D_{\mathrm{KL}} \left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \right) + \log(\mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)) \right) + \lambda_2 \right) \tag{B.30}$$

In order to be a maxima, the variation in $\mathcal{L}[p]$ for any variation in $p(\mathbf{z})$ must be zero.

$$\delta\mathcal{L} = \int \mathrm{d}\mathbf{z} \left(-(1+\lambda_{KL})(\log p + 1) + (\lambda_{KL} \left(D_{\mathrm{KL}} \left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \right) + \log(\mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)) \right) + \lambda_2) \right) \delta p \tag{B.31}$$

The du Bois-Reymond lemma implies the bracketed term must be identically zero.

$$0 = -(1+\lambda_{KL})(\log p + 1) + \left(\lambda_{KL}\left(D_{\mathrm{KL}}\left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|\right) + \log(\mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right))\right) + \lambda_2\right) \tag{B.32}$$

$$\log p = \frac{\lambda_{KL}}{1+\lambda_{KL}} D_{\mathrm{KL}}\left(\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|\right) - 1 + \frac{\lambda_{KL}}{1+\lambda_{KL}} \log(\mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)) + \frac{\lambda_2}{1+\lambda KL} \tag{B.33}$$

$$p = C(\lambda_{KL}, \lambda_2)\mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)^{\frac{\lambda_{KL}}{1+\lambda_{KL}}} \tag{B.34}$$

The second Lagrange variable, λ_2, will enforce normality; therefore, we may write simply

$$p(\mathbf{z}) \propto \mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)^{\alpha} \tag{B.35}$$

where $\alpha = \frac{\lambda_{KL}}{1+\lambda_{KL}}$ is introduced for brevity. This may be simplified if we consider the form of the normal distribution.

$$\begin{aligned} p(\mathbf{z}) &\propto \mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)^{\alpha} && \text{(B.36)} \\ &\propto \exp\left(-\frac{1}{2}\mathbf{z}^{\top}\mathbf{R}'^{-1}\mathbf{z}\right)^{\alpha} && \text{(B.37)} \\ &\propto \exp\left(-\frac{1}{2}\alpha\mathbf{z}^{\top}\mathbf{R}'^{-1}\mathbf{z}\right) && \text{(B.38)} \\ &\propto \exp\left(-\frac{1}{2}\mathbf{z}^{\top}(\frac{1}{\alpha}\mathbf{R}')^{-1}\mathbf{z}\right) && \text{(B.39)} \\ &\propto \mathcal{N}\left(\mathbf{0}, \frac{1}{\alpha}\mathbf{R}'\right) && \text{(B.40)} \end{aligned}$$

The Kullback-Liebler divergence between two zero-mean multivariate normal distributions is

$$D_{\mathrm{KL}}\left(\mathcal{N}\left(\mathbf{0}, \mathbf{R}\right) \| \mathcal{N}\left(\mathbf{0}, \mathbf{R}'\right)\right) = \frac{1}{2}\left(\mathrm{tr}\left(\mathbf{R}'^{-1}\mathbf{R}\right) - d - \log\left(\frac{\det \mathbf{R}}{\det \mathbf{R}'}\right)\right) \tag{B.41}$$

where d is the dimension of the distribution. Our constraint enforces that $D_{\mathrm{KL}}\left(p(\mathbf{z})\,\|\mathcal{N}\left(\mathbf{0},\mathbf{R}'\right)\right) = D_{\mathrm{KL}}\left(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|\right)$. Therefore,

$$D_{\mathrm{KL}}\left(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|\right) = D_{\mathrm{KL}}\left(\mathcal{N}\left(\mathbf{0},\frac{1}{\alpha}\mathbf{R}'\right)\|\mathcal{N}\left(\mathbf{0},\mathbf{R}'\right)\right) \tag{B.42}$$

$$= \frac{1}{2}\left(\mathrm{tr}\left(\mathbf{R}'^{-1}\frac{1}{\alpha}\mathbf{R}'\right) - d - \log\left(\frac{\det\frac{1}{\alpha}\mathbf{R}'}{\det\mathbf{R}'}\right)\right) \tag{B.43}$$

$$= \frac{d}{2}(\frac{1}{\alpha} - 1 - \log\frac{1}{\alpha}) \tag{B.44}$$

Rearranging, we have

$$-\exp\left(2\frac{D_{\mathrm{KL}}\left(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|\right)}{d}+1\right) = -\frac{1}{\alpha}\exp(-\frac{1}{\alpha}) \tag{B.45}$$

This cannot be solved for α in terms of elementary functions. The solution can be written in cloased form in terms of the Lambert W function.

$$\alpha = -W\Bigg(-\exp\left(\frac{2}{d}D_{\mathrm{KL}}\left(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|\right)+1\right)\Bigg) \tag{B.46}$$

This function has several interesting properties; it is normalizable, finite for $\boldsymbol{\phi}=\boldsymbol{\phi}'$, and decays monotonically as $D_{\mathrm{KL}}\left(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|\right)$ increases. Thus, by choosing $D_{\mathrm{KL}}\left(\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|\right)$ carefully, we may recover any kernel we choose.

Bibliography

[1] Gabriel Agamennoni, Juan I. Nieto, and Eduardo M. Nebot. An outlier-robust kalman filter. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, page 15511558, 2011. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5979605`.

[2] Andrea Censi. An accurate closed-form estimate of ICP's covariance. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, April 2007.

[3] Ienkaran Arasaratnam and Simon Haykin. Cubature kalman filters. *Automatic Control, IEEE Transactions on*, 54(6):12541269, 2009. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4982682`.

[4] Aleksandr Y. Aravkin and James V. Burke. Smoothing dynamic systems with state-dependent covariance matrices. *arXiv preprint arXiv:1211.4601*, 2012. URL `http://arxiv.org/abs/1211.4601`.

[5] A. Bachrach, S. Prentice, R. He, and N. Roy. RANGE - robust autonomous navigation in GPS-denied environments. *Journal of Field Robotics*, 28(5):644–666, September 2011.

[6] Abraham Bachrach, Ruijie He, and Nicholas Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217228, 2009. URL `http://multi-science.metapress.com/index/80586KML376K2711.pdf`.

[7] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation.* John Wiley and Sons, Inc, 2001.

[8] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164171, 1970. URL `http://www.jstor.org/stable/10.2307/2239727`.

[9] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3):433466, September 1995. ISSN 0360-0300. doi: 10.1145/212094.212141. URL `http://doi.acm.org/10.1145/212094.212141`.

[10] Ola Bengtsson and Albert-Jan Baerveldt. Robot localization based on scan-matchingestimating the covariance matrix for the IDC algorithm. *Robotics and Autonomous Systems*, 44(1):29–40, July 2003. ISSN 0921-8890. doi: 10.1016/S0921-8890(03)00008-3. URL `http://www.sciencedirect.com/science/article/pii/S0921889003000083`.

[11] Mauro Brenna. *Scan matching covariance estimation and SLAM: models and solutions for the scanSLAM algorithm*. PhD thesis, Artificial Intelligence and Robotics Laboratory Politecnico di Milano, 2009.

[12] Adam Bry, Abraham Bachrach, and Nicholas Roy. State estimation for aggressive flight in gps-denied environments using onboard sensing. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, page 18, 2012. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6225295`.

[13] E. De Castro and C. Morandi. Registration of translated and rotated images using finite fourier transforms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):700703, 1987. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4767966`.

[14] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, page 138, 1977. URL `http://www.jstor.org/stable/10.2307/2984875`.

[15] Arthur Gelb. *Applied optimal estimation*. MIT Press, May 1974. ISBN 9780262570480.

[16] Jerry D. Gibson, Boneung Koo, and Steven D. Gray. Filtering of colored noise for speech enhancement and coding. *Signal Processing, IEEE Transactions on*, 39 (8):17321742, 1991. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=91144`.

[17] Roland Goecke, Akshay Asthana, Niklas Pettersson, and Lars Petersson. Visual vehicle egomotion estimation using the fourier-mellin transform. In *Intelligent Vehicles Symposium, 2007 IEEE*, page 450455, 2007. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4290156`.

[18] Neil J. Gordon, David J. Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, page 107113, 1993. URL `http://digital-library.theiet.org/content/journals/10.1049/ip-f-2.1993.0015`.

[19] Andrew C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1990. ISBN 0521405734.

[20] Huy Tho Ho and Roland Goecke. Optical flow estimation using fourier mellin transform. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, page 18, 2008. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4587553`.

[21] Berthold KP Horn and Brian G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1):185203, 1981. URL `http://www.sciencedirect.com/science/article/pii/0004370281900242`.

[22] Congwei Hu, Wu Chen, Yongqi Chen, and Dajie Liu. Adaptive kalman filtering for vehicle navigation. *Journal of Global Positioning Systems*, 2(1):4247, 2003. URL `http://www.gmat.unsw.edu.au/wang/jgps/v2n1/v2n1pf.pdf`.

[23] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Proceedings of the International Symposium of Robotics Research (ISRR)*, Flagstaff, AZ, 2011.

[24] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

[25] R. E. Kalman. A new approach to linear filteringand prediction problems. *Transactions of the ASMEJournal of Basic Engineering*, 82 (Series D):35–45, 1960.

[26] Kwang Hoon Kim, Jang-Gyu Lee, and Chan-Gook Park. Adaptive two-stage extended kalman filter for a fault-tolerant INS-GPS loosely coupled system. *IEEE Transactions on Aerospace and Electronic Systems*, 45(1):125–137, 2009. ISSN 0018-9251. doi: 10.1109/TAES.2009.4805268.

[27] Jonathan Ko and Dieter Fox. GP-BayesFilters: bayesian filtering using gaussian process prediction and observation models. *Auton. Robots*, 27(1):7590, July 2009. ISSN 0929-5593. doi: 10.1007/s10514-009-9119-x. URL `http://dx.doi.org/10.1007/s10514-009-9119-x`.

[28] Jonathan Ko, Daniel J. Klein, Dieter Fox, and Dirk Haehnel. *GP-UKF: Unscented Kalman Filters with Gaussian Process Prediction and Observation Models.*

[29] Michael L. Littman, Richard S. Sutton, and Satinder Singh. Predictive representations of state. *Advances in neural information processing systems*, 14: 15551561, 2001. URL `http://www-2.cs.cmu.edu/Groups/NIPS/NIPS2001/papers/psgz/CN12.ps.gz`.

[30] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333349, 1997. URL `http://link.springer.com/article/10.1023/A%3A1008854305733`.

[31] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, 1981. URL `http://www.ri.cmu.edu/pub_files/pub3/lucas_bruce_d_1981_1/lucas_bruce_d_1981_1.ps.gz`.

[32] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: incremental smoothing and mapping. *IEEE Trans. on Robotics, TRO*, 24(6):1365–1378, December 2008.

[33] Raman K. Mehra. On the identification of variances and adaptive kalman filtering. *IEEE Transactions on Automatic Control*, AC-15:175–184, 1970.

[34] Arman Melkumyan and Fabio Ramos. Multi-kernel gaussian processes. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 1408–1413, 2011.

[35] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[36] E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, January 1964. ISSN 0040-585X, 1095-7219. doi: 10.1137/1109020. URL `http://epubs.siam.org/doi/abs/10.1137/1109020`.

[37] Philippe Naveau, Marc G. Genton, and Xilin Shen. A skewed kalman filter. *Journal of Multivariate Analysis*, 94(2):382400, 2005. URL `http://www.sciencedirect.com/science/article/pii/S0047259X04001150`.

[38] K. Paliwal and Anjan Basu. A speech enhancement method based on kalman filtering. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, volume 12, page 177180, 1987. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1169756`.

[39] B. Srinivasa Reddy and B. N. Chatterji. An FFT-based technique for translation, rotation, and scale-invariant image registration. *Image Processing, IEEE Transactions on*, 5(8):12661271, 1996. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=506761`.

[40] M. Stakkeland, O. Overrein, E.F. Brekke, and O. Hallingstad. Tracking of targets with state dependent measurement errors using recursive BLUE filters. In *12th International Conference on Information Fusion, 2009. FUSION '09*, pages 2052–2061, 2009.

[41] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Mit Press, 2005. ISBN 9780262201629.

[42] Geoffrey S. Watson. Smooth regression analysis. *Sankhy: The Indian Journal of Statistics, Series A*, page 359372, 1964. URL `http://www.jstor.org/stable/10.2307/25049340`.

[43] Andrew Gordon Wilson and Zoubin Ghahramani. Generalised wishart processes. *Uncertainty in Artificial Intelligence*, 2011.