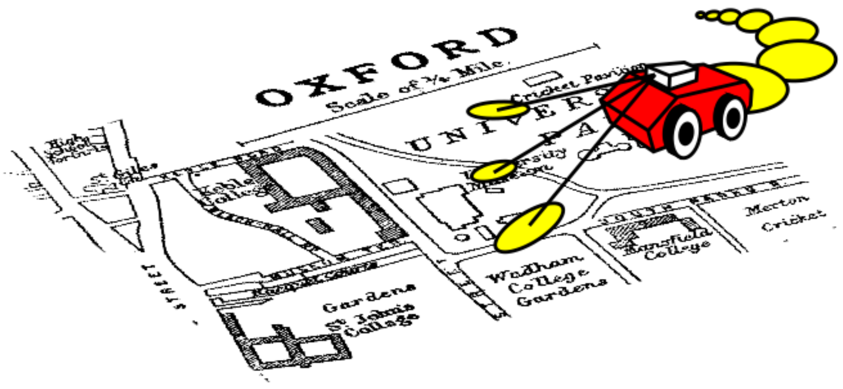
Consistency and SLAM



Simon J. Julier
Department of Computer Science
University College London
Malet Place
London WC1E 6BT, UK
`S.Julier@cs.ucl.ac.uk`

August 13, 2006

Contents

Contents **i**

Preface **1**

I Introduction 1

1 The Structure and Properties of Kalman Filter-Based SLAM **3**
1.1 Structure of SLAM 3
1.2 Properties of SLAM 4

II Modelling Errors and SLAM 7

2 Process and Observation Noise Errors **9**
2.1 A Simplified Mathematical Framework for Modelling Errors 9
2.2 The Effects of Modelling Errors on SLAM 9
2.2.1 Prediction 10
2.2.2 Example 11
2.3 Summary 14

3 Mitigating the Effects of Modelling Errors **15**
3.1 Locking the Map 15
3.2 Particle Filter Based Methods 15
3.3 Build New Local Maps 16

III Linearisation Errors and SLAM 17

4 Linearisation and Other Kinds of Approximations **19**
4.1 Mapping From A Stationary AGV 19
4.2 A Concrete Example 20
4.2.1 Behavior of a Stationary Vehicle 20
4.2.2 The Behavior of a Moving Vehicle 22

5 Mitigating the Effects of Linearisation Errors **25**
5.1 Using Higher Order Transformations to Represent Uncertainty 25
5.2 Eliminating Effects Using Angle Sensors 25
5.2.1 Ideal “Compass” 25
5.2.2 Real “Compasses” 25
5.3 Eliminating Effects using Batch Solutions 26
5.4 Eliminating Effects Using Reparameterisations 26
5.5 Eliminating Effects Using Submapping 26

6 Conclusions **27**

A System Models **29**
A.1 Process Model: Fundamental Bicycle Process Model 29
A.2 Linear Displacement Observation Model 29
A.2.1 Linear Observation Model 29
A.3 Range-Bearing Observation Model 29
A.4 Compass Observation Model 30

B The Mathematics of Modelling Errors **31**
B.0.1 Modelling Error Framework 31
B.0.2 The Effects of Modelling Errors 31

Index **36**

Preface

As this summer school attests, SLAM is a vibrant research topic of great practical importance. The need to know where something is can be vital for many types of operations. Much research has been focussed on the question of methods to implement SLAM. However, many of these discussions have missed a fundamental point: does SLAM actually *work*? This question is not as simple as it seems because there are many defitions of work which could be adopted here.

Unlike many other classes of tracking and estimation problems, SLAM is not fully observable. Rather, it is detectable. The implications of this are extremely important. In normal tracking applications errors are "washed out". Process noise makes the system stochastically controllable (the correlation structure is such that any state can be modified by the observations) and stochastically observable (any state can be reconstructed from the observations). However, in SLAM once errors are in the system they cannot be removed again. This raises several important questions:

1. **Data Association Errors.** What happens when you associate a beacon incorrectly?

2. **Modelling Errors.** What are the impacts of unmodelled system processes on SLAM? All process models, for example, contain errors which lead to correlated noise terms. How do these impact the performance of the filter?

3. **Probability Transformation Errors.** Any real system is nonlinear, and so the probability densities must undergo nonlinear transformations. In most cases these can only be computed approximately. As a result, approximations creep in.

The first issue has been addressed a number of different authors as part of the operation of the filter and therefore we do not consider it further in this tutorial. Rather, we consider the remaining issues which have received limited, but increasing attention. In particular, we consider the following: how much harder does the problem become when one moves from a tracking problem with targets in known locations with targets in unknown locations.

This tutorial restricts itself to Kalman filter-based SLAM. Although is not the only means of addressing the SLAM problem, it is one of the most widely used. Because looking at the detail of specific algorithms and do not want to obscure the wood from the trees, will specifically look at small, controlled examples.

A brief summary is as follows. We show that the former case is okay as long as we stick to the usual consistency definitions. The latter case represents a significant problem for SLAM. Given the latter is more important, we then spend some time looking at different potential solutions. Although these help to reduce the onset of problems, they do not eliminate them altogether.

This tutorial is decomposed into three parts. The first part provides a survey of SLAM and revisits some of its properties. Although readers are probably sick to death of this by now, we use the opportunity to describe our notation and revisit some aspects of SLAM which are relevant for the analysis. The second part considers the problem with process and observation models. It shows that errors in the models do not, direclty, contribute to issues with the models given a mean squared sense. However, there could be issues with outliers and nonlinearities. We describe a couple of solutions to these problems. The fourth section wraps up with some conclusions.

Part I

Introduction

Chapter 1

The Structure and Properties of Kalman Filter-Based SLAM

1.1 Structure of SLAM

The Full Covariance (FC-SLAM) formulation of the SLAM problem was developed by the seminal papers from Smith, Self and Cheeseman [25]. The environment is assumed to be populated by a set of landmarks that can be detected by sensors fixed to a platform.

The state space for a map of N landmarks is

$$\mathbf{x}(k) = \left[\mathbf{x}_v(k)\ \mathbf{p}_1^T(k)\ \ldots\ \mathbf{p}_N^T(k)\right]^T$$

where $\mathbf{x}_v(k)$ is the vehicle state and $\mathbf{p}_i(k)$ is the location of the ith landmark. The mean and covariance of this estimate are

$$\hat{\mathbf{x}}(k \mid k) = \left[\hat{\mathbf{x}}_v^T(k \mid k) \ldots \hat{\mathbf{p}}_N^T(k \mid k)\right]^T \tag{1.1}$$

$$\mathbf{P}(k \mid k) = \begin{pmatrix} \mathbf{P}_{vv}(k \mid k) & \mathbf{P}_{v1}(k \mid k) & \ldots & \mathbf{P}_{vN}(k \mid k) \\ \mathbf{P}_{1v}(k \mid k) & \mathbf{P}_{11}(k \mid k) & \ldots & \mathbf{P}_{1N}(k \mid k) \\ \mathbf{P}_{2v}(k \mid k) & \mathbf{P}_{21}(k \mid k) & \ldots & \mathbf{P}_{2N}(k \mid k) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{Nv}(k \mid k) & \mathbf{P}_{N1}(k \mid k) & \ldots & \mathbf{P}_{NN}(k \mid k) \end{pmatrix} \tag{1.2}$$

where $\mathbf{P}_{vv}(k \mid k)$ is the covariance of the vehicle's position estimate, $\mathbf{P}_{ii}(k \mid k)$ is the covariance of the position estimate of the ith landmark and $\mathbf{P}_{ij}(k \mid k)$ is the cross-correlation between the estimate of landmarks i and j.

By assumption, all of the landmarks are stationary, so there is no accumulation of process noise. Thus, the process model takes the form

$$\mathbf{f}\left[\mathbf{x}_N(k-1), \mathbf{u}(k-1), \mathbf{v}(k-1)\right] = \begin{pmatrix} \mathbf{f}_v\left[\mathbf{x}_v(k-1), \mathbf{u}(k-1), \mathbf{v}(k-1)\right] \\ \mathbf{0} \end{pmatrix}$$

where $\mathbf{u}(k-1)$ is the control input and $\mathbf{v}(k-1)$ is the process noise. Similarly, the process noise covariance matrix is

$$\mathbf{Q}(k) = \begin{bmatrix} \mathbf{Q}_v(k) & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix}. \tag{1.3}$$

The observation model for the vehicle's observation of the ith landmark is

$$\mathbf{z}_i(k) = \mathbf{h}_i\left[\mathbf{x}_v(k), \mathbf{p}_i(k), \mathbf{w}(k)\right] \tag{1.4}$$

where $\mathbf{w}(k)$ is the observation noise with covariance $\mathbf{R}(k)$. Whenever the vehicle observes a landmark which is in the map, the joint system is updated using the Kalman filter update equations.

The Kalman filter update equations are

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\boldsymbol{\nu}(k) \tag{1.5}$$

$$\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}(k)^T \tag{1.6}$$

where $\boldsymbol{\nu}(k)$ is the innovation vector, $\mathbf{S}(k)$ is its covariance, and $\mathbf{W}(k)$ is the Kalman gain. These values are computed by

$$\boldsymbol{\nu}(k) = \mathbf{z}(k) - \hat{\mathbf{z}}(k \mid k-1) \tag{1.7}$$

$$\hat{\mathbf{z}}(k \mid k-1) = \mathbf{h}_i\left[\hat{\mathbf{x}}_v(k \mid k-1), \hat{\mathbf{p}}_i(k \mid k-1), \mathbf{0}\right] \tag{1.8}$$

$$\mathbf{C}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}^T(k) \tag{1.9}$$

$$\mathbf{S}(k) = \mathbf{H}(k)\mathbf{C}(k) + \mathbf{R}(k) \tag{1.10}$$

$$\mathbf{W}(k) = \mathbf{C}(k)\mathbf{S}(k)^{-1} \tag{1.11}$$

and $\mathbf{H}(k)$ is the Jacobian of $\mathbf{h}_i[\cdot,\cdot,\cdot]$ evaluated around $\hat{\mathbf{x}}(k \mid k-1)$ (hence $\hat{\mathbf{x}}_v(k \mid k-1)$ and $\hat{\mathbf{p}}_i(k \mid k-1)$).

Later we exploit the structure that for many types of observation models (e.g., range-only, bearing-only or range-bearing sensors) the Jacobian matrix derived from the linearized observation model for most types of sensors can be written in partitioned form as:

$$\boldsymbol{\nabla}\mathbf{h}_i = \begin{bmatrix} -\boldsymbol{\nabla}_x\mathbf{h}_i & \boldsymbol{\nabla}_p\mathbf{h}_i \end{bmatrix} \tag{1.12}$$

where the negative sign on $\boldsymbol{\nabla}_p\mathbf{h}_i$ denotes the fact that the sensor observes the difference between the beacon and vehicle positions. When the vehicle re-observes a beacon that has already been initialised and placed in the map, the joint vehicle and beacon state estimate is updated using a normal Kalman update expression. When the vehicle observes a beacon that it has not seen before, a new beacon estimate is created and inserted into the map. To initialise a beacon position estimate, the inverse of the beacon observation equation is used[1]:

$$\mathbf{p}_i(k) = \mathbf{g}_i\left[\mathbf{x}_v(k), \mathbf{z}_i(k), \mathbf{w}(k)\right].$$

We require the condition that the estimate is *consistent*. In other words,

$$\mathbf{P}(i \mid j) - \mathrm{E}\left[\tilde{\mathbf{x}}(i \mid j)\tilde{\mathbf{x}}^T(i \mid j)\right] \geq \mathbf{0}. \tag{1.13}$$

where $\tilde{\mathbf{x}}(i \mid j) = \mathbf{x}(i) - \hat{\mathbf{x}}(i \mid j)$ is the error in the estimate and $\geq \mathbf{0}$ means that the result is positive semidefinite[2].

Traditional SLAM analysis has observed that there is an issue with scalability: the computational costs involved are extremely high and need to be reduced. However, an important issue which less frequently considered is the problem of when does and doesn't SLAM work? In particular, what's not been looked at are issues such as modelling errors and effects of linearisation.

1.2 Properties of SLAM

Based on commonly accepted optimality properties of the Kalman filter, it may appear that the above formulation is rigorous and optimal. A number of studies have been made of map building systems [42169421694216942169] of this type under the assumption that the process and observation models are linear. These studies conclude the following:

1. *SLAM is a detectable estimation problem.* Because the sensors are only capable of measuring the location of beacons relative to the vehicle's position, map building provides no information about the *absolute* position of the vehicle or of the beacons in the map. The only information about the absolute position is specified in the initial conditions. One consequence is that at no point in time can the vehicle covariance $\mathbf{P}_{vv}(k \mid k)$ be *smaller* than the initial covariance $\mathbf{P}_{vv}(0 \mid 0)$, i.e., $\mathbf{P}_{vv}(k \mid k) - \mathbf{P}_{vv}(0 \mid 0)$ is positive semidefinite.

[1]When the beacon position is not observable from a single observation (e.g., the sensor measures only range or bearing), this equation can be generalised to be the inverse of a *sequence* of observations [17].

[2]Bar-Shalom also stipulates that the error is zero-mean [2]. However, because this requires the unrealistic assumptions that there are no errors in the process models, the observation models, or biases casused be linearisation and other simplifications, we prefer our more general definition.

2. *The covariance of the beacons is non-increasing.* Because the beacons are stationary and no process noise is injected into them, their covariances only change as the result of a beacon update. Since an update never increases the covariance, the beacon covariance can only remain the same or be reduced. This property is extremely important. Given that the whole system is non-observable, non-increasing (by some measure) beacon covariances are necessary to ensure that the vehicle and beacon covariances remain finite

3. *In the limit, the structure of the map becomes uniform.* The estimated errors in all of the beacon covariances will eventually become the same and perfectly correlated with one another. The lower bound of the covariance is a function of the vehicle and observation covariances at the time the first beacon is initialized in the map and the process noise that is subsequently injected.

Csorba proved that the limit function, given the observation model of the form (1.12), is [4]

$$\mathbf{P}\left(k \mid k-1\right) = \mathbf{P}_0\left(k \mid k-1\right) + \mathbf{P}_E\left(k \mid k-1\right) \tag{1.14}$$

where the second term has the structure

$$\mathbf{P}_E\left(k \mid k-1\right) = \begin{bmatrix} \mathcal{A}_\mathcal{E} & \mathcal{B}_\mathcal{E} & \dots & \mathcal{B}_\mathcal{E} \\ \vdots & \vdots & \ddots & \vdots \\ {\mathcal{B}_\mathcal{E}}^T & \mathcal{C}_\mathcal{E} & \dots & \mathcal{C}_\mathcal{E} \end{bmatrix} \tag{1.15}$$

The matrices $\mathcal{A}_\mathcal{E}$, $\mathcal{B}_\mathcal{E}$ and $\mathcal{C}_\mathcal{E}$ are of the form

$$\mathcal{A}_\mathcal{E} = \begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 & 0 & \dots & 0 \\ \sigma_{yx}^2 & \sigma_{yy}^2 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

$$\mathcal{B}_\mathcal{E} = \begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}$$

$$\mathcal{C}_\mathcal{E} = \begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{bmatrix}$$

These matrices are only a function of the position estimates of the vehicles and the beacons.

Given this form, Csorba proved that whenever the system is updated with *any* sensor whose observation model can be written in the form of Equation 1.12, the updated covariance can be written as

$$\mathbf{P}\left(k \mid k\right) = \mathbf{P}_0\left(k \mid k\right) + \mathbf{P}_E\left(k \mid k-1\right)$$

where $\mathbf{P}_0\left(k \mid k\right)$ is positive semidefinite and $\mathbf{P}_0\left(k \mid k\right) \leq \mathbf{P}_0\left(k \mid k-1\right)$. In other words, $\mathbf{P}_E\left(k \mid k-1\right)$ defines a "lower bound" on the covariance matrix — the covariance can never be reduced below this value. Intuitively, this term reflects the fact that the observations provide information about the relative displacement between the vehicle and the beacon. They do not provide information about the absolute location of the map in a global coordinate system. Therefore, any SLAM algorithm which affects the steady-state performance of SLAM system will affect the value of $\mathbf{P}_E\left(k \mid k-1\right)$.

The undetectability comes from uncertainty about the original location. One means of overcoming this is through assuming that the vehicle starts at the origin and this fact is known perfectly. This principle is used in many sub-mapping algorithms.

The discussion so far has assumed that the filter operates perfectly. In particular,

1. The measurements are associated perfectly. When the filter makes an update, the identity of the beacon has to be known. It is assumed that this is known. There has been a significant body of work on SLAM. Much of this is related to the so-called "loop-closing" problem: a platform moves in a circuit and returns to an earlier part of the map. The fact that the vehicle has returned must be recognised and can be used to update the map characteristics.

2. The models are all perfect. In the context here perfection means that the noises really are independent[3]. Although analyses have been carried out for tracking of objects individually, little consideration has been paid in SLAM.

3. The first order approximations in the filter are adequate. Again, most presentations of SLAM assume that the filter, because it is derived from the Kalman filter, simply "works".

However, how realistic is it to assume that these conditions hold true in practice? as explained earlier these conditions do not hold true in practice. Therefore, an important question is what properties above hold true in the "real" situation? In this paper, however, we are interested in a more basic question: is the filter even consistent?

Of the three, the first condition has received the most attention

As discussed later, this condition means that the filter designer's favourite friend — stabilising noise [20] — cannot be used to compensate for all difficulties encountered with SLAM.

It is important to note that the above analysis was carried out looking at the linearised propagation equations of the Kalman filter. The analysis does not consider how the errors *actually* propagate through the Kalman filter equations. Furthermore, these analyses are often made under the assumption that results which hold true for linear systems can be extrapolated to nonlinear systems. This is not generally the case.

[3]The often quoted assumption of Gaussianity is *not* needed for a Kalman filter.

Part II

Modelling Errors and SLAM

Chapter 2

Process and Observation Noise Errors

Modelling errors cover everything about models which aren't fully accounted for.

There are many causes of errors. These include simplifications of dynamics, or unknown inputs such as wind on a UAV [16] or hand movement [23]. Although these are all nonlinear systems, in this chapter we only consider linear systems. As explained in Appendix **??**, the vehicle state is assumed

To separate the discussion from what happens in the next section, we only consider linear models.

2.1 A Simplified Mathematical Framework for Modelling Errors

Errors in the process noise must be compensated for by the well-known procedure of "tuning" (or enlarging) $\mathbf{Q}(k)$, the process noise covariance matrix, until the filter estimates are consistent [20]. Implicitly, this tuning procedure assumes that the residual unmodelled dynamics are stationary and temporally uncorrelated. However, this assumption is incorrect (the residual dynamics are temporally correlated) and this subsection derives an expression which shows how large the additional process noise must be.

Because the system is linear, the true system (**??**) can be written as[1]

$$\mathbf{x}(k) = \mathbf{F}(k-1)\,\mathbf{x}(k-1) + \mathbf{v}(k)\,. \tag{2.1}$$

However, the filter uses an approximate model of the form

$$\mathbf{x}(k) = \hat{\mathbf{F}}(k-1)\,\mathbf{x}(k-1)\,. \tag{2.2}$$

The observation and inverse observation models for the ith beacon are

$$\mathbf{z}_i(k) = \mathbf{H}_i(k)\,\mathbf{x}(k) + \mathbf{w}(k) \tag{2.3}$$

$$\mathbf{p}(k) = 111 \tag{2.4}$$

In Appendix **??**, we show that the effects of modelling errors can be accounted for by a suitable definition of the process noise. However it should be noted that, because of the structure of the SLAM problem, the structure of the process noise is limited to that given in (1.3): the noise can *only* be injected into the vehicle state and not into the entire state space.

2.2 The Effects of Modelling Errors on SLAM

We analyse the effects of modelling errors by considering the following scenario: both the true system and the filter evolve using the same system model up to timestep $k-1$. At that point a "fault" occurs: the true model changes while the filter continues to use its nominal model[2]. One example where this could occur is if the system has switched dynamics where the system can operate in a set of discrete modes.

[1] Control inputs are neglected for simplicity. If there is an error in the model for how control inputs enter the system, these must be included as another error term.

[2] We describe this as a "fault" because fault detection algorithms, in reality, attempt to detect the case when the model in the filter ceases to accurately describe the behaviour of the system.

2.2.1 Prediction

Consider first the case of a system which has been correctly modelled up to timestep k. At this instance, the filter experiences a fault. Rather than predict according to (2.2), the system actually evolves according to (2.1). In the case that there is tracking and the map is known already, $\mathbf{Q}(k)$ can be chosen to guarantee consistency.

Theorem 1. *When the system experiences a fault at time step k, no finite value of* $\mathbf{Q}(k)$ *of the form in (1.3) exists to maintain filter consistency in time step* $k+1$.

Proof. The true system evolves according to (2.1). Therefore, the true system state at k is

$$\mathbf{x}(k) = \mathfrak{F}(k)\,\mathfrak{x}(k-1) + \mathfrak{v}(k-1)\,. \tag{2.5}$$

However, the nominal system evolves according to (2.2). Therefore,

$$\hat{\mathbf{x}}(k \mid k-1) = \mathbf{F}(k)\,\hat{\mathbf{x}}(k-1 \mid k-1)\,. \tag{2.6}$$

Subtracting (2.5) from (2.6), the *actual* prediction error is

$$\begin{aligned}\tilde{\mathbf{x}}(k \mid k-1) &= \hat{\mathbf{x}}(k \mid k-1) - \mathfrak{x}(k)\\ &= \mathbf{F}(k)\,\hat{\mathbf{x}}(k-1 \mid k-1) - \mathfrak{F}(k)\,\mathfrak{x}(k-1) - \mathfrak{v}(k-1)\\ &= \mathbf{F}(k)\,\tilde{\mathbf{x}}(k-1 \mid k-1) + \tilde{\mathbf{F}}(k)\,\mathfrak{x}(k-1) - \mathfrak{v}(k-1)\end{aligned} \tag{2.7}$$

where

$$\tilde{\mathbf{F}}(k) = \mathbf{F}(k) - \mathfrak{F}(k)$$

is the error in the process model.

The filter predicts the covariance to be

$$\mathbf{P}(k \mid k-1) = \begin{pmatrix} \mathbf{F}_v(k)\,\mathbf{P}_{vv}(k \mid k-1)\,\mathbf{F}_v^T(k) + \mathbf{Q}(k) & \mathbf{P}_{v1}(k \mid k)\,\mathbf{F}_v^T(k) & \dots & \mathbf{P}_{vN}(k \mid k)\,\mathbf{F}_v^T(k) \\ \mathbf{F}_v(k-1)\,\mathbf{P}_{1v}(k \mid k) & \mathbf{P}_{11}(k \mid k) & \dots & \mathbf{P}_{1N}(k \mid k) \\ \mathbf{F}_v(k-1)\,\mathbf{P}_{2v}(k \mid k) & \mathbf{P}_{21}(k \mid k) & \dots & \mathbf{P}_{2N}(k \mid k) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{F}_v(k-1)\,\mathbf{P}_{Nv}(k \mid k) & \mathbf{P}_{N1}(k \mid k) & \dots & \mathbf{P}_{NN}(k \mid k) \end{pmatrix} \tag{2.8}$$

However the actual mean squared error, calculated by taking the outer product and expectation of (2.7), is

$$\mathfrak{P}(k|k-1) = \begin{pmatrix} \mathfrak{P}_{vv}(k|k-1) & \mathfrak{P}_{v1}(k|k) & \dots & \mathfrak{P}_{vN}(k|k) \\ \mathfrak{P}_{1v}(k|k) & \mathbf{P}_{11}(k \mid k) & \dots & \mathbf{P}_{1N}(k \mid k) \\ \mathfrak{P}_{2v}(k|k) & \mathbf{P}_{21}(k \mid k) & \dots & \mathbf{P}_{2N}(k \mid k) \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{P}_{Nv}(k|k) & \mathbf{P}_{N1}(k \mid k) & \dots & \mathbf{P}_{NN}(k \mid k) \end{pmatrix} \tag{2.9}$$

where

$$\begin{aligned}\mathfrak{P}_{vv}(k|k-1) = {} & \mathbf{F}_v(k)\,\mathbf{P}_{vv}(k-1 \mid k-1)\,\mathbf{F}_v^T(k) + \mathbf{F}_v(k)\,\mathrm{E}\left[\tilde{\mathbf{x}}(k-1 \mid k-1)\,\mathfrak{x}^T(k-1)\right]\tilde{\mathbf{F}}_v^T(k)\\ & + \tilde{\mathbf{F}}_v(k)\,\mathrm{E}\left[\mathfrak{x}(k-1)\,\tilde{\mathbf{x}}^T(k-1 \mid k-1)\right]\mathbf{F}_v^T(k)\\ & + \tilde{\mathbf{F}}_v(k)\,\mathrm{E}\left[\mathfrak{x}(k-1)\,\mathfrak{x}^T(k-1)\,k-1\right]\tilde{\mathbf{F}}_v^T(k) + \mathbf{Q}(k)\end{aligned} \tag{2.10}$$

$$\mathfrak{P}_{vi}(k|k-1) = \mathbf{P}_{v1}(k \mid k-1)\,\mathbf{F}_v^T(k) + \mathrm{E}\left[\tilde{\mathbf{p}}_i(k \mid k-1)\,\mathfrak{x}^T(k)\right]\tilde{\mathbf{F}}_v^T(k) \tag{2.11}$$

In other words, there are additional error coupling terms due to the correlations between the estimation errors and the *true* state of the system. In the timestep that the error first occurs, the expected value of the error is zero. In consequence, the effect of the modelling error is to increase the process noise by the amount $\tilde{\mathbf{F}}_v(k)\,\mathrm{E}\left[\mathfrak{x}(k-1)\,\mathfrak{x}^T(k-1)\,k-1\right]\tilde{\mathbf{F}}_v^T(k)$ which can be accounted for by increasing $\mathbf{Q}(k)$ appropriately. However, the difficulty arises with the cross correlation term. This shows that the effect of the modelling error is to add a correlation term between the error in the beacon estimate and the true vehicle position state. Prior to the modelling error the expected error is zero and so nothing gets included. Post it, however, correlations can arise.

Consider the update. The observation is given by

$$\mathbf{z}(k) = \mathfrak{H}(k)\,\mathfrak{x}(k) + \mathfrak{w}(k).$$

However, the filter predicts it to be

$$\hat{\mathbf{z}}(k \mid k-1) = \mathbf{H}(k)\,\hat{\mathbf{x}}(k \mid k-1).$$

Assuming that there are no observation model errors then $\mathbf{H}(k) = \mathfrak{H}(k)$ and the update is given by

$$\begin{aligned}\hat{\mathbf{x}}(k \mid k) &= \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\,\mathbf{H}(k)\,(\mathfrak{x}(k) - \hat{\mathbf{x}}(k \mid k-1)) + \mathbf{W}(k)\,\mathfrak{w}(k)\\ &= \hat{\mathbf{x}}(k \mid k-1) - \mathbf{W}(k)\,\mathbf{H}(k)\,\tilde{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\,\mathfrak{w}(k)\end{aligned}$$

Therefore, the error in the estimate is

$$\begin{aligned}\tilde{\mathbf{x}}(k \mid k) &= \hat{\mathbf{x}}(k \mid k-1) - \mathfrak{x}(k)\\ &= (\mathbf{I} - \mathbf{W}(k)\,\mathbf{H}(k))\,\tilde{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\,\mathfrak{w}(k)\\ &= (\mathbf{I} - \mathbf{W}(k)\,\mathbf{H}(k))\Big(\mathbf{F}(k)\,\tilde{\mathbf{x}}(k-1 \mid k-1) + \tilde{\mathbf{F}}(k)\,\mathfrak{x}(k-1) - \mathfrak{v}(k-1)\Big)\end{aligned}$$

If $\mathbf{Q}(k)$ is chosen such that the prediction is consistent then, for the first update step, the update is consistent as well. However, the significance of the update is that it includes a component of the true state value, thus causing a correlation between the vehicle state and the true state to arise. If the update causes the values of the beacons to change, the errors in the beacon estimates become correlated with the true state.

Putting this back into the earlier result, we see that this causes the correlation to change in an unmodelled way. Therefore, the filter will fail.

□

The above theory shows that there is a qualitative difference between the non-SLAM and SLAM problem. In the non-SLAM problem, when a modelling error occurs process noise can be added to the entire state space to ensure that the filter remains consistent. However, in SLAM process noise cannot be added to the beacon estimates. As a result, *any* corruption of the vehicle-beacon cross correlation structure leads to an estimate which ceases to be consistent.

One method of trying to avoid this is to add some slack to the beacons by making them conservative.

Theorem 2. *If $\mathbf{P}(k \mid k)$ is consistent and $\mathbf{R}(k)$ is consistent, then a SLAM filter with a freshly initialised beacon will be consistent as well.*

Proof. Take outer products and demonstrate the result. □

Therefore, this suggests that it is possible to provide a "buffer" in the filter before the modelling errors arise. Two strategies can be taken here: inflate $\mathbf{Q}(k)$ so that $\mathbf{P}(k \mid k)$ is larger and inflate $\mathbf{R}(k)$ so that the beacon initialisation is larger. However, these effects are indirect at best. The steady-state structure of SLAM given in (**??**) means that the observation noise will get filtered off of the beacons eventually. The analysis of the monobot stuff shows that greatly increasing the process noise decreases the worst case performance of the filter: basically the error in the initial estimate cannot be filtered off very aggresively.

2.2.2 Example

The above result can be illustrated in the following simple example. The vehicle state merely consists of the vehicle position $x(k)$ and velocity $\dot{x}(k)$. There is a single beacon $p_1(k)$. The true system can operate in one of two modes. In the first mode it exhibits a damped oscillation and the continuous time process model is

$$\hat{\mathbf{F}}_c^1(k) = \begin{pmatrix} 0 & 1 \\ -\omega & -\zeta \end{pmatrix}$$

where $\omega = 1.1$ and $\zeta = 0.11$. In the second mode, the filter simply behaves like a constant nominal velocity filter and so

$$\hat{\mathbf{F}}_c^2(k) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

In both cases the observation noise is injected as a constant acceleration[3]

Because the filter is tightly tuned, $\hat{\mathbf{Q}}(k)$ is chosen according to (**??**). This guarantees that the predicted position sub-block is consistent.

The following filters were tested:

1. **No SLAM; true model.** This filter assumes that the beacons are known perfectly and the true model is known at each time step. This represents the optimal filter performance. Furthermore, the normalised value should be 2.

2. **No SLAM; nominal model.** This filter assumes that the beacons are known perfectly but utilizes the nominal value for the process model. As such, it validates (**??**). Again, the normalised value should be 2.

3. **SLAM; true model; tightly tuned.** This filter utilizes SLAM and the correct process model at each stage and the normalised value should be 3.

4. **SLAM; nominal model; tightly tuned.** This filter utilizes SLAM and the nominal process model at each state with (**??**). Again, the normalised value should be 3.

The results of these filters are plotted in Figure 2.1 for a 100 time step scenario. The true filter operates in mode 1 between timesteps 1 and 20. From timesteps 21 to 80 it operates in model 2 and then switches back to mode 1 again for the remaining 20 timesteps. The diagram shows that, unsurprisingly, SLAM or tracking using the correct process model guarantees a consistent estimate. The diagram also shows that, in the tracking mode, (**??**) leads to a consistent estimate as well. In fact, when the modelling errors begin, the process noise increases and this, in turn, causes the mean squared error to decrease. However, the same is not true for the SLAM filter: its normalised staete error rises from 3 to 3.3, indicating that the filter is inconsistent. The normalised state error slowly reduces, but it does not reach its original value by the time the simulation run ceases.

Although the results in this section suggest that the impact of modelling errors are minor, it should be noted that the fundamental system is linear and thus not particularly prone to catastrophic failures. Nonetheless, the results are somewhat worrying. One possible reason for this is the tightly tuned nature of the filter. Prior to the modelling error, the filter was tuned precisely. This lead to the sub-block associated with the beacons in (**??**) to be $\mathbf{0}$ to be inconsistent. A more realistic assumption is that the filter will be tuned conservatively so that the beacon estimates will contain some slack. In particular,

Theorem 3. *If the filter is tuned to yield a consistent prediction, the initialised beacon will be consistent as well.*

Proof. The new beacon is initialised according to

$$\hat{\mathbf{x}}_{N+1}(k \mid k) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mathbf{G}_v & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{x}_v(k)\, k-1 \\ \mathbf{p}(k)\, k-1 \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_w \end{pmatrix} \tag{2.12}$$

□

Therefore, two additional models were investigated:

1. **SLAM; true model; loosely tuned.** Same as above, but now the noise is scaled.

2. **SLAM; nominal model; loosely tuned.** Same as above, but now the noise is scaled.

[3]To derive the discrete-time matrices used in (2.1) and (2.2), the transformation

$$\hat{\mathbf{F}}^i(k) = \exp\left\{\hat{\mathbf{F}}_c^i(k)\,\Delta t\right\}$$

is applied. The process noise is calculated by applying the convolution integral over the prediction step. Therefore, ...

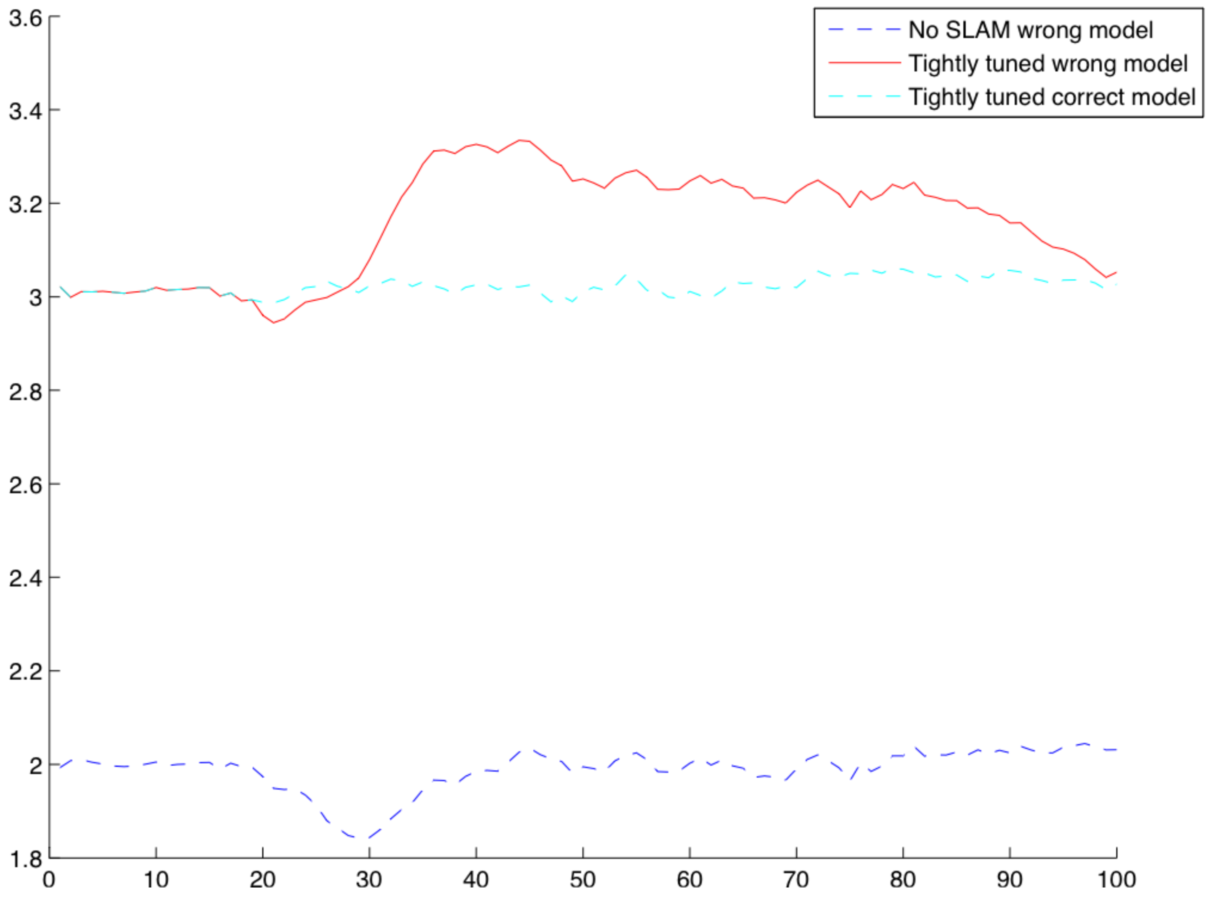


Figure 2.1: The effects of process model errors on the normalised state error of tightly tuned SLAM filters.

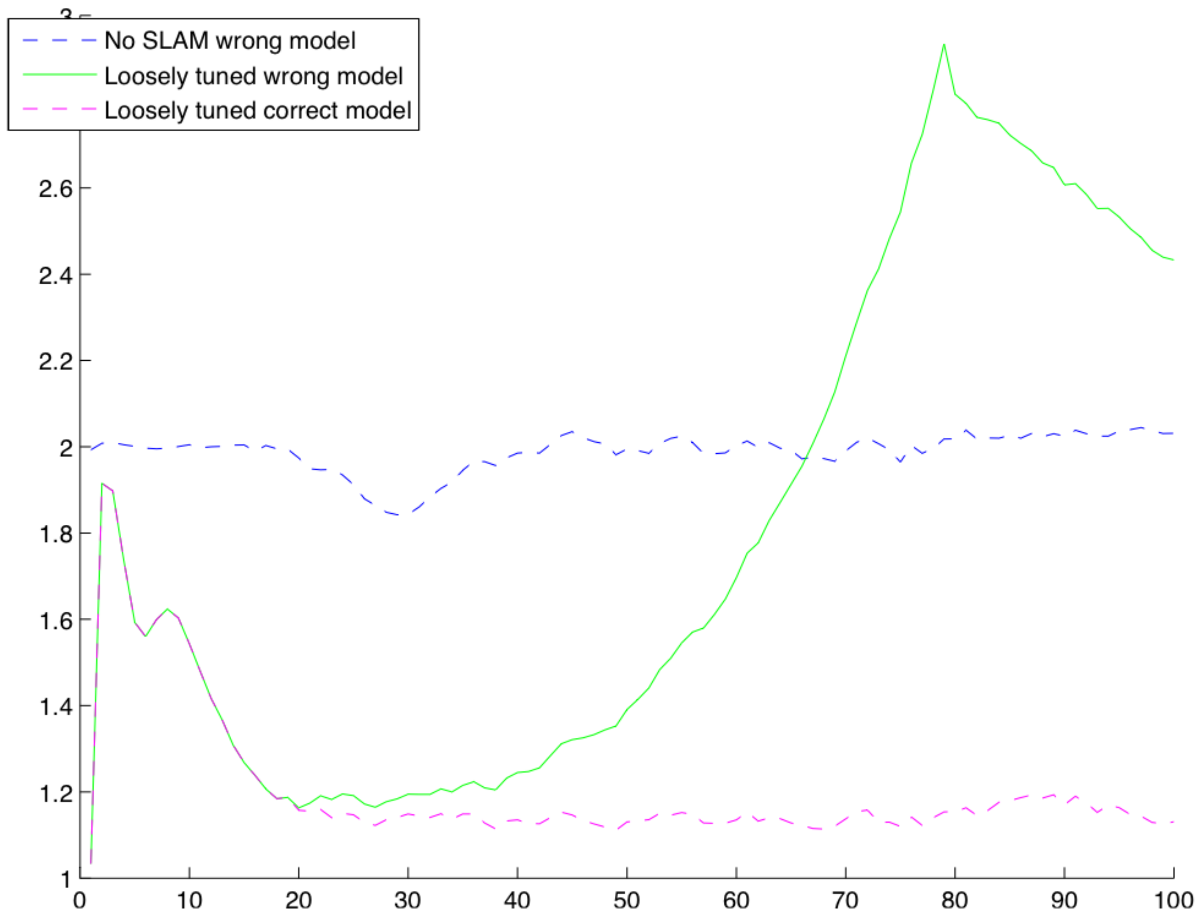


Figure 2.2: The effects of process model errors on the normalised state error of tightly tuned SLAM filters.

Figure 2.2 shows the results when the filter is "loosely" tuned by scaling the process noise up by a factor $\alpha = 600$. The results show that the filter can be made consistent. However, as can be seen, the normalised state error rises through time: only the fact that the true system reverts to the original behaviour that the error drops sharply.

2.3 Summary

In this chapter we have considered the effects of modelling errors on SLAM. We have shown that, even in the single beacon case, modelling errors behave qualitatively very differently from those in the non-SLAM case. Furthermore, the tuning will depend on a run-by-run basis of what beacons are initialised in which order and when.

Chapter 3

Mitigating the Effects of Modelling Errors

The last chapter has shown that modelling errors can have a significant impact on the performance of a filter. Qualitatively we can think of two types of errors: those which cause a slow drift, and those which are caused by sudden unexpected movements. However, the analysis in the last section shows that, irrespective of the type of error, the effect is the same: correlations between the vehicle state and the beacons (2.11) cause the filter to cease to be inconsistent. Therefore, the strategy needs to be such that the vehicle-beacon estimates cannot become correlated with one another.

3.1 Locking the Map

A first strategy would be to stop the map estimates from updating [12] using a Schmidt-Kalman Filter [24].

Most of the computational burden of the KF arises from the fact that the optimal weight causes all of the states to be updated. If only a subset of the states were updated, optimality can be traded against convenient computational properties. For example, suppose the vehicle observes the ith beacon, and only the vehicle and the ith beacon are to be updated. This is achieved if

$$\mathbf{W}(k) = \begin{bmatrix} \mathbf{W}_v(k) \\ \mathbf{0} \\ \mathbf{W}_i(k) \\ \mathbf{0} \end{bmatrix} \tag{3.1}$$

is used instead of $\mathbf{K}(k)$ in (**??**). Taking outer products and expectations, the covariance is

$$\begin{aligned} \mathbf{P}(k \mid k) &= \mathbf{X}\mathbf{P}(k \mid k-1)\mathbf{X}^T \\ &\quad + \mathbf{W}(k)\mathbf{R}(k)\mathbf{W}^T(k) \end{aligned} \tag{3.2}$$

where

$$\mathbf{X} = \mathbf{I} - \mathbf{W}(k)\boldsymbol{\nabla}^T\mathbf{h}_i.$$

The results are shown in Figure **??**.

However, an important difficulty with this approach is that most stochastic fault detection algorithms take a number of time steps to identify when faults occur (and it's not clear how well these algorithms would work with SLAM).

3.2 Particle Filter Based Methods

Extremely accurate representation of map using full nonlinear representation. However, limitations of FastSLAM are widely appreciated.

Pupilli and Calway [23] propose a method which combine particle filters with Unscented Kalman Filters (UKFs). Basic idea is two-fold: first, maintain multiple hypotheses (to work out where the map plugs back) and second suspend map building.

3.3 Build New Local Maps

Idea here is to generalise previous method. If you can't work out where the map fits, simply build and hope to plug it back together later. Similar to kidnapped robot problem with visual saliency. Need some kind of search / association algorithm to align things, which would be a pain to find....

Part III

Linearisation Errors and SLAM

Chapter 4

Linearisation and Other Kinds of Approximations

4.1 Mapping From A Stationary AGV

Consider the following scenario. An AGV is placed at an unknown location in its environment with a specified mean $\hat{\mathbf{x}}_v\left(0 \mid 0\right)$ and covariance $\mathbf{P}_{vv}\left(0 \mid 0\right)$. The AGV then uses a sensor (such as a laser range finder) to measure the position of the beacons relative to the AGV. The AGV is assumed to remain stationary, so no process noise is injected as beacons are initialised and updated. Because the AGV's state is unchanging, and because beacon positions are only measured relative to the AGV, the AGV's position estimate should not change. Therefore, $\hat{\mathbf{x}}_v\left(k \mid k\right) = \hat{\mathbf{x}}_v\left(0 \mid 0\right)$ for all timesteps k. This condition is satisfied if the following theorem holds:

Theorem 4. *If an AGV estimate is initialised with a non-zero covariance, beacon estimates are initialised using (**??**) and (**??**), and all observation covariances are finite, then the state estimate of the AGV will remain unchanged if and only if*

$$\boldsymbol{\nabla}_x \mathbf{h}_1 - \boldsymbol{\nabla}_p \mathbf{h}_1 \, \boldsymbol{\nabla} \mathbf{g}_1^x = \mathbf{0} \tag{4.1}$$

for all timesteps k.

Proof. Assume that the beacon is initialised at timestep 0. We prove (4.1) by considering the first two timesteps.

The beacon estimate is initialised into the map at timestep 0. Therefore, the state of the system with the intialised beacon is given by (**??**) and (**??**). Because the vehicle is stationary and no process noise is injected, the predicted state of the vehicle at timestep 1 is $\hat{\mathbf{x}}_1\left(1 \mid 0\right) = \hat{\mathbf{x}}_1\left(0 \mid 0\right)$ and $\mathbf{P}_1\left(1 \mid 0\right) = \mathbf{P}_1\left(0 \mid 0\right)$.

From (**??**), a necessary and sufficient condition to ensure that the vehicle estimate does not change is that the Kalman weight (gain matrix) which is applied to it should be $\mathbf{0}$. Therefore, the weight should be of the form

$$\mathbf{W}\left(k\right) = \begin{bmatrix} \mathbf{W}_v\left(k\right) \\ \mathbf{W}_p\left(k\right) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{W}_p\left(k\right) \end{bmatrix},$$

where the dimension of $\mathbf{0}$ is the same as $\hat{\mathbf{x}}_v\left(k \mid k\right)$ and $\mathbf{W}_p\left(k\right)$ is the weight applied to the beacon. From (**??**), the condition is equivalent to demanding that

$$\mathbf{P}\left(k \mid k-1\right) \boldsymbol{\nabla}^T \mathbf{h}_1^x = \begin{bmatrix} \mathbf{0} \\ \mathbf{W}_p \mathbf{S}\left(k\right) \end{bmatrix}, \tag{4.2}$$

where the second term is the weight applied to the beacon (not considered here). The actual weight applied to the vehicle is

$$\mathbf{W}_v\left(1\right) = \mathbf{P}_v\left(1 \mid 0\right) \left(\boldsymbol{\nabla}_x \mathbf{h}_1 - \boldsymbol{\nabla}_p \mathbf{h}_1 \, \boldsymbol{\nabla} \mathbf{g}_1^x\right)^T \mathbf{S}^{-1}\left(1\right). \tag{4.3}$$

Because $\mathbf{P}_v\left(1 \mid 0\right)$ and $\mathbf{S}^{-1}\left(1\right)$ are nonsingular, a necessary and sufficient condition for $\mathbf{W}_v\left(1\right)$ to be $\mathbf{0}$ is

$$\mathbf{0} = \boldsymbol{\nabla}_x \mathbf{h}_1 - \boldsymbol{\nabla}_p \mathbf{h}_1 \, \boldsymbol{\nabla} \mathbf{g}_1^x.$$

This result proves the theorem for a single timestep. This can be readily extended to an arbitrary number of timesteps. □

The importance of this result is that the behavior of the vehicle and beacon depends critically on the Jacobian matrices used to initialise the beacon. In the special case that the observation model is linear and time invariant[1] ($\nabla_x \mathbf{h}_i = \mathbf{H}^x, \nabla_p \mathbf{h}_i = \mathbf{H}^p$ and $\nabla \mathbf{g}_i^x = \mathbf{G}^x$), the condition requires that $\mathbf{H}^p \mathbf{G}^x = \mathbf{H}^x$. However, in a general nonlinear system where the Jacobian matrices are functions of noisy observations and erroneous estimates, it is not clear that the condition in (4.1) can be guaranteed to hold. Furthermore, because this condition is a *structural* relationship, normal tuning procedures (such as inflating the observation noise covariances) cannot circumvent the problem. If the vehicle estimate changes for one value of $\mathbf{R}(k)$, it will change for all (finite) values of $\mathbf{R}(k)$. In fact, as we now show, this condition cannot be guaranteed even for the simplest case of a position-orientation AGV model and a range-bearing sensor model.

4.2 A Concrete Example

Consider the following simple system. The vehicle state is described by its position (x_v, y_v) and orientation θ_v in some global coordinate system. The vehicle is equipped with a sensor that is able to return the range r and bearing ϕ of a target relative to the sensor platform. The sensor is a rotating scanner (such as a laser range finder) which completes one revolution per second.

We now consider two special cases for this system — a stationary vehicle with no process noise, and a vehicle which moves in a circle with nominally constant control inputs.

4.2.1 Behavior of a Stationary Vehicle

First consider the special case that the vehicle is stationary, there is only one beacon, and *no process noise* acts on the system. In this situation the process model reduces to the identity matrix and

$$\mathbf{f}_v \left[\mathbf{x}_v (k-1), \mathbf{0}, \mathbf{0}\right] = \mathbf{x}_v (k-1) .$$

where the dimension of $\mathbf{0}$ is the same as $\hat{\mathbf{x}}_v (k \mid k)$. At timestep 0 the beacon is initialised into the map and the mean and covariance are set using (**??**) and (**??**). This case can be analysised by Theorem 4. Assuming that (4.2) has been obeyed up to timestep k, the numerator of $\mathbf{W}(k)$ is

$$\begin{pmatrix} \mathbf{P}_n (k \mid k) & \mathbf{P}_n (k \mid k) \nabla^T \mathbf{g}_n^x \\ \nabla \mathbf{g}_n^x \mathbf{P}_n (k \mid k) & \nabla \mathbf{g}_n^x \mathbf{P}_n (k \mid k) \nabla^T \mathbf{g}_n^x + \mathbf{A} \end{pmatrix} \begin{pmatrix} \begin{bmatrix} \mathbf{h}_i^{x_v y_v T} \\ \mathbf{h}_i^{\phi T} \end{bmatrix} \\ \mathbf{h}_i^{x_i y_i T} \end{pmatrix}$$

where $\mathbf{A}(k)$ is a positive semidefinite matrix corresponding to the partially filtered observation noise. Therefore, the weight on the vehicle $\mathbf{W}_v$ is

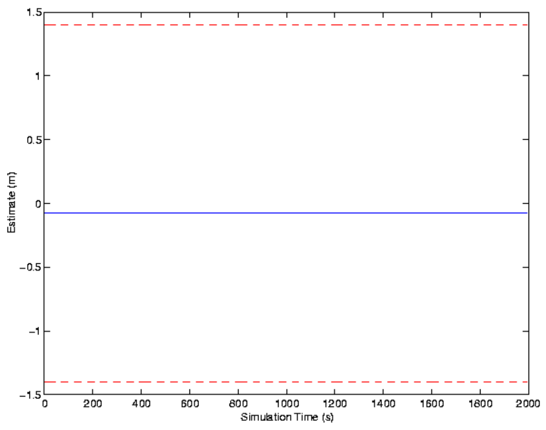
$$\mathbf{W}_v = \mathbf{P}_n (k \mid k) \left\{ \begin{bmatrix} \mathbf{h}_i^{x_v y_v T} \\ \mathbf{h}_i^{\theta T} \end{bmatrix} + \begin{bmatrix} \mathbf{g}_i^{x_v y_v T} \\ \mathbf{g}_i^{\theta T} \end{bmatrix} \mathbf{h}_i^{x_i y_i T} \right\} .$$

Substituting from (A.7) to (A.9) and using the property that $\mathbf{h}_i^{x_v y_v T} = -\mathbf{h}_i^{x_i y_i T}$,

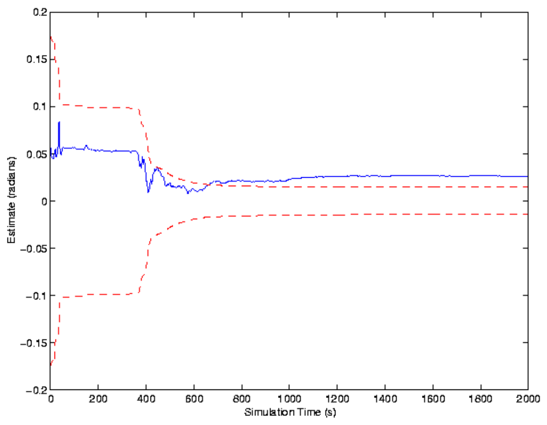
$$\mathbf{W}_v = \mathbf{P}_n (k \mid k) \begin{bmatrix} \mathbf{0} \\ \mathbf{h}_i^{\theta T} - \mathbf{g}_i^{\theta T} \mathbf{h}_i^{x_v y_v T} \end{bmatrix} .$$

Therefore, the weight on the vehicle position states is always guaranteed to be $\mathbf{0}$. However, this is *not* the case for the vehicle orientation state. Substituting from Equations A.3, A.8 and A.2, it can be readily shown that this weight is non-zero only if the angle between the vehicle and the beacon $(\theta_v + \phi)$ is the *same* as the value when the beacon was first initialised. It should be emphasised that these analytical results reflect a fundamental failure in the structure of the cross correlation between the vehicle and beacon estimates. The errors occur irrespective of the magnitude of covariances and they are not the result of subtle numerical implementation errors.

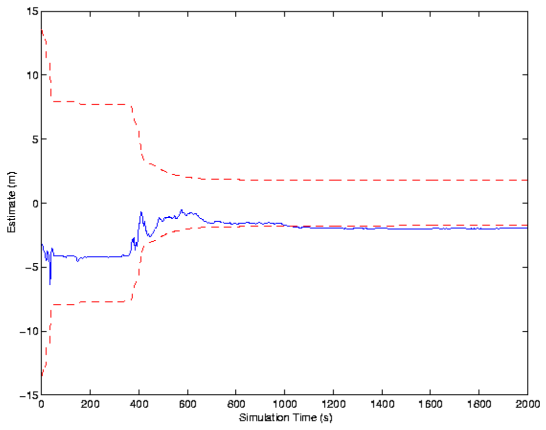
[1]Even with nonlinear process and observation models, such a system structure arises in the relative map [5] and geometric projection filter [21] which decouple the problem of map building from vehicle localization. However, with these approaches it appears that the map can only be used as an aid to beacon gating and it cannot be used to update the vehicle position estimate.



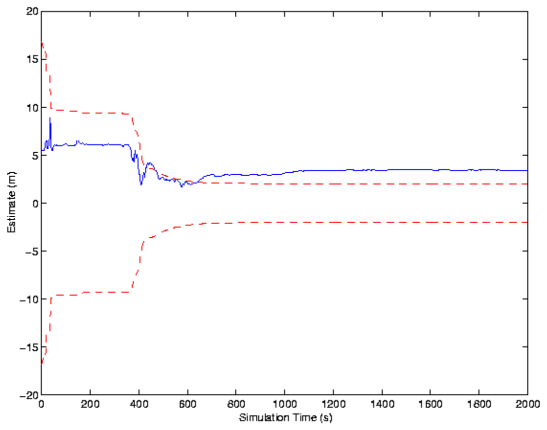
(a) Error in x_v.



(b) Error in θ_v.



(c) Error in x_1.



(d) Error in y_1.

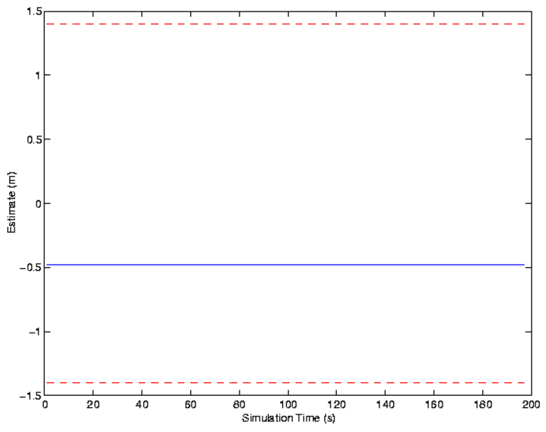
Figure 4.1: Estimation errors and 2 standard deviation bounds. The standard deviation bounds are shown as dashed lines.

This behavior can be clearly seen in a simulation study of this scenario. For the results in this paper we use the following conditions. The vehicle initially starts at the origin with the standard deviations in x_v and y_v of 0.7m and θ_v of $5°$. It observes a beacon at $(97.89, 70.1)$ with an accurate sensor whose observation noise covariance is $\mathbf{R}(k) = \text{diag}(0.25\text{m}^2, (1°)^2)$. Figure 4.1 plots the time history of the estimates of x_v, θ_v, x_1 and y_1. As expected, the estimate of x_v does not change. However, the estimate of θ_v immediately starts to change and its covariance begins to decline. By the end of the run, the orientation covariance is less than 6% of its initial value. However, this update is entirely spurious — there is no additional information about the vehicle's orientation and so the orientation estimate becomes inconsistent. In turn, the beacon estimate becomes inconsistent as well. Extending the simulation further shows that, in the limit, the steady-state covariance of the beacon estimate does not become 0 but is a value greater than the initial vehicle position covariance.

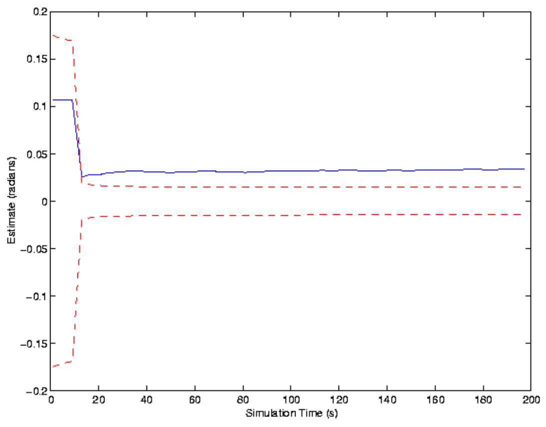
The value of $(\theta_v + \phi)$ also changes if the vehicle and beacon configuration changes. Figure 4.2 shows the result when the vehicle, at timestep 10, "teleports" to the position $(50, 50)$. This jump occurs instantaneously and without uncertainty, i.e., at the same time the estimate changes, the true vehicle position changes by the same amount[2]. As can be seen the results are dramatic. Although the error in the estimate of x_v remains unchanged after the instantaneous translation, observations of the beacons lead to a large spurious reduction in both the vehicle orientation and the beacon position estimates. Again, these reductions correspond to inconsistent estimates.

It must be emphasised that the inconsistencies in both examples are not simply due to the fact that the obser-

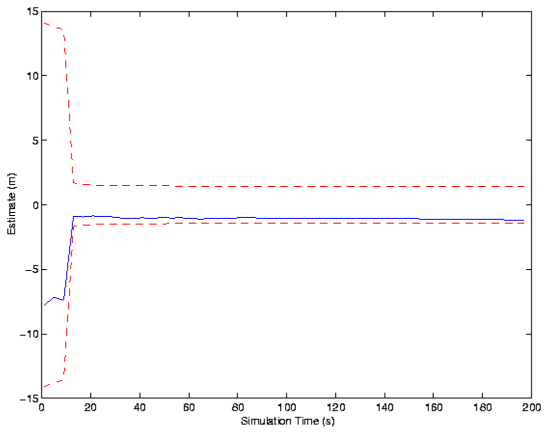
[2]This is equivalent to the process model $x_v(k+1) = x_v(k) + \Delta x$, $y_v(k+1) = y_v(k) + \Delta y$ where Δx and Δy are known.



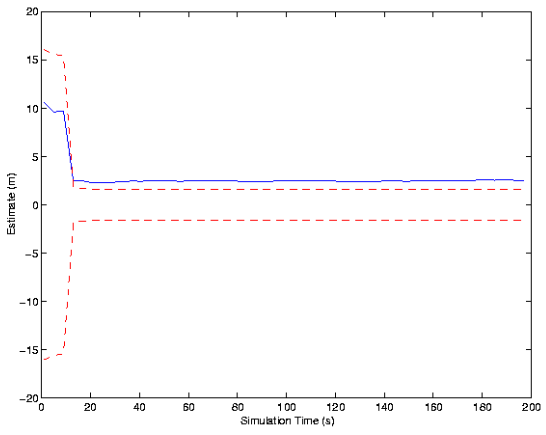
(a) Error in x_v.



(b) Error in θ_v.



(c) Error in x_1.



(d) Error in y_1.

Figure 4.2: Estimation errors and 2 standard deviation bounds. For a vehicle which "teleports" from position $(0, 0)$ to $(50, 50)$ at timestep 10.

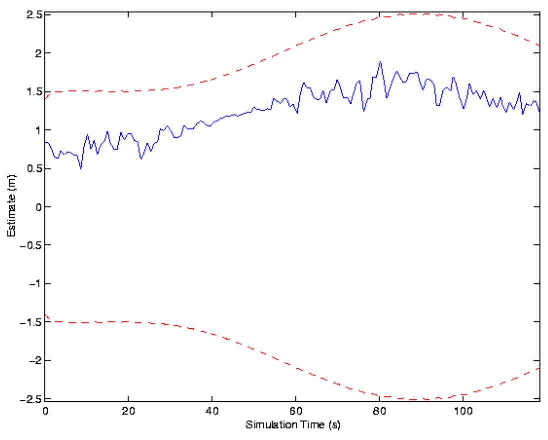
vation Jacobian is calculated using the noise-corrupted sensor observations. Even if the *true* state of the beacon and the vehicle were always used to calculate the Jacobian of the observation equation, *any* motion by the vehicle that affects the observation Jacobian will lead to inconsistency analogous to a violation of the condition of Equation 4.1 in the stationary case described in Section **??**. In fact, a perfectly known change in the AGV's orientation – *with no change in position* – will have the same effect. In the next section we generalise the example to include the accumulation of process noise by a moving vehicle.

4.2.2 The Behavior of a Moving Vehicle

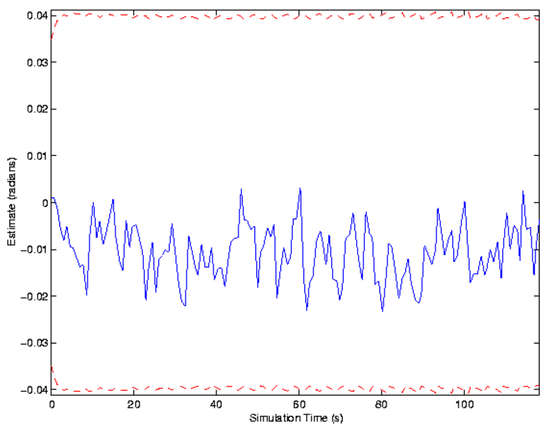
In this section we consider the case of a moving vehicle in a long-duration SLAM simulation. As in our example of a stationary vehicle with a range-bearing sensor, our goal is to keep the scenario as simple as possible to demonstrate the perniciousness of the inconsistency problem. To this end, we assume that the vehicle travels in a circle with constant control inputs $\mathbf{u}(k) = [1\ 2°]$ and process noise standard deviations $\mathbf{Q}(k) = \text{diag}\{0.25, (0.3°)^2\}$. The vehicle observes an environment which contains 5 beacons.

The time history of the first 600 timesteps of the vehicle and beacon estimates are shown in Figure 4.3 and Figure 4.4 respectively. The errors in the beacons and the beacon estimates appear to have stabilised within the two sigma bounds and it appears that this algorithm is performing in a satisfactory manner.

Figures 4.5 and 4.6 show the time histories when the experiment is allowed to run for 16,000 timesteps. These

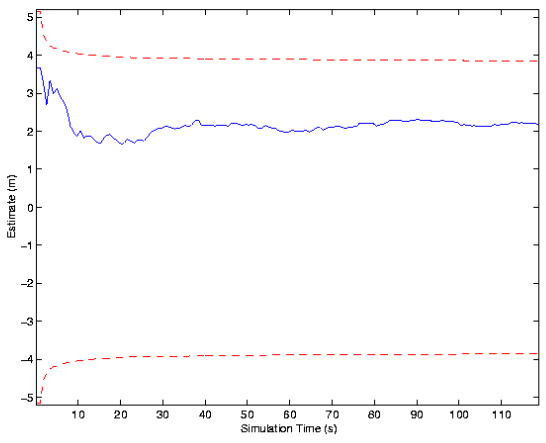


(a) Error in x_v.

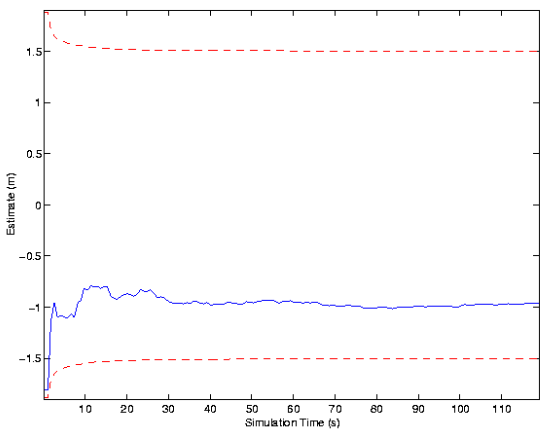


(b) Error in θ_v.

Figure 4.3: Results for the first 600 seconds (3000 timesteps) of a moving vehicle.



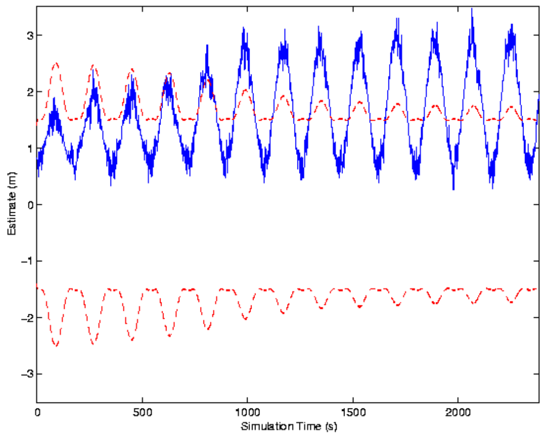
(a) Error in x_1.



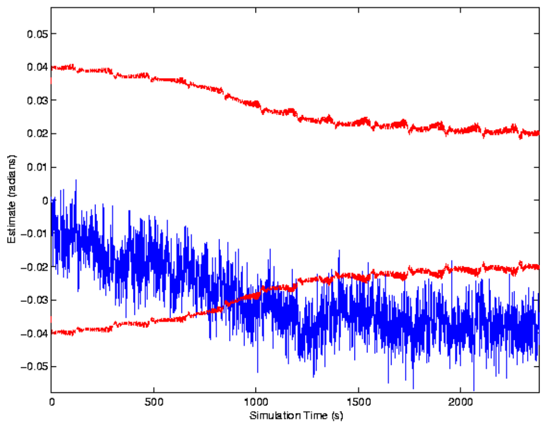
(b) Error in y_1.

Figure 4.4: Results for the first 600 seconds (3000 timesteps) of beacton 1.

results clearly show that the apparent consistency is only a short-term phenomena: within less than five thousand time steps (1000 updates per beacon) the map has become inconsistent. We speculate that this behavior has not been recognised in the literature for two main reasons. First, most systems reported in the literature (such as [3] or [5]) only present results for the first few hundred timesteps. Over such short durations the signs of divergence are not very prominent. Second, the few long-duration studies (such as [18]) have all used compasses to measure the absolute orientation of the vehicle. A compass causes the orientation errors to be filtered out, and significantly reduces the rate of divergence.

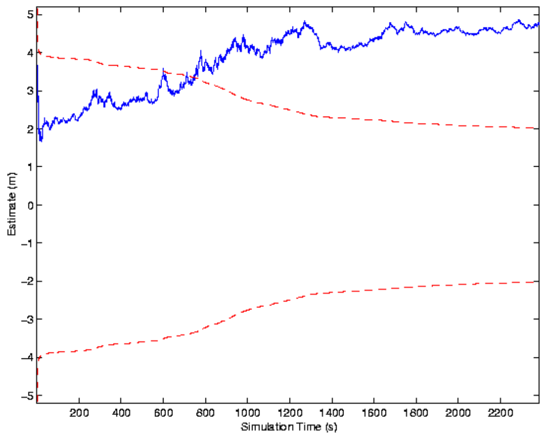


(a) Error in x_v.

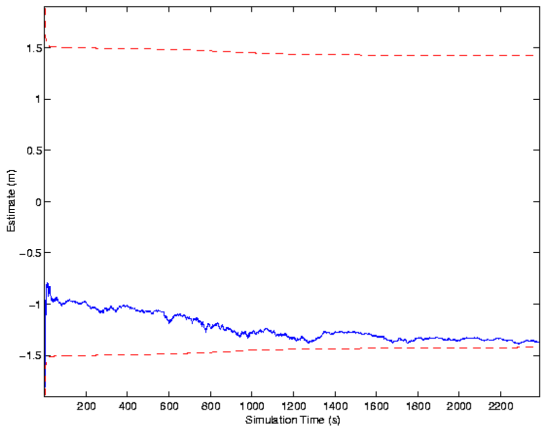


(b) Error in θ_v.

Figure 4.5: Results for the first 3200 seconds (16000 timesteps).



(a) Error in x_1.



(b) Error in y_1.

Figure 4.6: Estimation errors and 2 standard deviation bounds for beacon 1 for 3200 seconds (16000 timesteps). The standard deviation bounds are the pairs of dashed lines.

Chapter 5

Mitigating the Effects of Linearisation Errors

5.1 Using Higher Order Transformations to Represent Uncertainty

The analysis has considered the issue of linearisation. However, the limitations of linearisation are widespread and are well known. In consequence, a number of higher order filters have been derived including the truncated second order filter, particle filters, and the unscented Kalman filter.

Particle filter implementation is best known for FastSLAM. This method uses particles to marginalise over the different states. Not only can it theoretically overcome linearisation errors but, because it maintains a description of the entire state space, it can represent much richer representations of the distribution. However, concerns have been raised about FastSLAM and, in particular, its tendency to diverge over time.

The most significant other implementations are to apply the Unscented Transformation. Andrade-Cetto [1] proposed using the UT within SLAM. However, the UT has some scaling issues associated with the dimensions of the state space (computational complexity and point scaling) and so they proposed to apply it to calculate the sub-block of the vehicle covariance only. Empirical results suggested improved performance. However, there is the difficutly that the cross-correlations are only approximated and consistency is not completely clear.

Martinez-Cantin [19] applies the UT to the entire SLAM problem.

5.2 Eliminating Effects Using Angle Sensors

Talk about the use of a compass or some similar device. Add illustration here of how well a compass does — or doesn't — perform. This could be a good illustration of modelling errors in action.

5.2.1 Ideal "Compass"

An ideal compass provides orientation errors without difficulty. Examples could be a compass which is calibrated in an electrically nice environment, sun sensors, or horizon sensors when the view is unobstructed, or even stuff involving vanishing points.

Point to show is that the correlation between the beacon estimates goes to 0.

5.2.2 Real "Compasses"

The magnetometers are strongly affected by local magnetic fields (so-called "hard iron biases"). Gebre-Egziabher [8] provides the following model. Let B_H be the earth's magnetic field (0.5 gauss), θ be the pitch angle, and the magnetic offsets due to local elements be δB_x and δB_y, then a compass measures the quantity

$$\theta_{INS} = \tan^{-1}\left(\frac{B_H \sin\theta_{true} + \delta B_y}{B_H \cos\theta_{true} + \delta B_x}\right) \tag{5.1}$$

5.3 Eliminating Effects using Batch Solutions

Discuss here how using multiple timesteps can help — potentially includes GraphSLAM.

5.4 Eliminating Effects Using Reparameterisations

Introduce and talk about robocentric mapping.

5.5 Eliminating Effects Using Submapping

Reduces overall angular uncertainty.

Chapter 6

Conclusions

This document has looked at the problem of linearisation errors in SLAM and, in particular, orientation errors. Have shown that these errors cause the filter to fail. Not only do these produce subtle numerical inconsistencies but great big hunking failures as well. Discussed a number of different strategies. Although these partially solve the problem, none of them eliminate the issue altogether.

Appendix A

System Models

This appendix presents the process and observation models used throughout the example in this paper. Although this material is standard and has been presented in many publications before, we include it here for completeness.

It should be noted that the *same* process model is used in all the examples. However, in the first section which considers modelling errors, we assume that the orientation is known perfectly.

A.1 Process Model: Fundamental Bicycle Process Model

The vehicle model is the standard equation for a steered bicycle [7]:

$$\mathbf{x}_v(k+1) = \begin{pmatrix} x_v(k) + V(k)\Delta t \cos(\delta(k) + \theta_v(k)) \\ y_v(k) + V(k)\Delta t \sin(\delta(k) + \theta_v(k)) \\ \theta_v(k) + \frac{V(k)\Delta t \sin(\delta(k))}{B} \end{pmatrix},$$

where the timestep is Δt, the control inputs are the wheel speed $V(k)$ and steer angle $\delta(k)$ and the vehicle wheel base is B. The process noises are additive disturbances which act on $V(k)$ and $\delta(k)$.

A.2 Linear Displacement Observation Model

A.2.1 Linear Observation Model

In Section 1.1 the vehicle uses a linear $(\Delta x, \Delta y)$ position sensor. Therefore,

$$\mathbf{h}_i\left[\mathbf{x}_v(k), \mathbf{p}_i(k), \mathbf{w}(k)\right] = \begin{bmatrix} \Delta x_i(k) \\ \Delta y_i(k) \end{bmatrix} = \begin{bmatrix} x_i(k) - x_v(k) + w_x(k) \\ y_i(k) - y_v(k) + w_y(k) \end{bmatrix}$$

A new landmark is initialized from

$$\mathbf{g}_i\left[\mathbf{x}_v(k), \mathbf{z}_i(k), \mathbf{w}(k)\right] = \begin{bmatrix} \Delta x_i(k) + x_v(k) \\ \Delta y_i(k) + y_v(k) \end{bmatrix}.$$

The observation covariance is

$$\mathbf{R}(k) = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix}$$

A.3 Range-Bearing Observation Model

The observation model is

$$\mathbf{h}_i\left[\mathbf{x}_v(k), \mathbf{p}_i(k), \mathbf{w}(k)\right] = \begin{bmatrix} \sqrt{(x_i - x_v)^2 + (y_i - y_v)^2} \\ \tan^{-1}\left(\frac{y_i - y_v}{x_i - x_v}\right) - \theta_v \end{bmatrix} \quad \text{(A.1)}$$

The Jacobian for this equation can be written as

$$\nabla \mathbf{h}_i^x = \begin{pmatrix} \mathbf{h}_i^{x_v y_v} & \mathbf{h}_i^{\phi} & \mathbf{h}_i^{x_i y_i} \end{pmatrix},$$

where

$$\mathbf{h}_i^{x_v y_v} = \begin{pmatrix} -(x_i - x_v)/r & -(y_i - y_v)/r \\ (y_i - y_v)/r^2 & -(x_i - x_v)/r^2 \end{pmatrix}, \tag{A.2}$$

$$\mathbf{h}_i^{\theta_v} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \tag{A.3}$$

$$\mathbf{h}_i^{x_i y_i} = -\mathbf{h}_i^{x_v y_v}. \tag{A.4}$$

The observation noise Jacobian is

$$\mathbf{h}_i^{w} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{A.5}$$

Let $\alpha(k) = \theta_v(k) + \phi(k)$. Inverting Equation A.1, the beacon position is initialised as

$$\mathbf{g}_i \left[\mathbf{x}_v\left(k\right), \mathbf{w}\left(k\right)\right] = \begin{bmatrix} x_v(k) + r(k)\cos[\alpha(k)] \\ y_v(k) + r(k)\sin[\alpha(k)] \end{bmatrix}. \tag{A.6}$$

and the Jacobian is

$$\nabla \mathbf{g}_i^x = \begin{pmatrix} \mathbf{g}_i^{x_v y_v} & \mathbf{g}_i^{\theta} \end{pmatrix},$$

where

$$\mathbf{g}_i^{x_v y_v} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{A.7}$$

$$\mathbf{g}_i^{\theta} = \begin{bmatrix} -r(k)\sin[\alpha(k)] \\ r(k)\cos[\theta_v(k) + \phi(k)] \end{bmatrix}, \tag{A.8}$$

$$\mathbf{g}_i^{w} = \begin{bmatrix} \cos[\alpha(k)] & -r(k)\sin[\alpha(k)] \\ \sin[\alpha(k)] & r(k)\cos[\alpha(k)] \end{bmatrix}. \tag{A.9}$$

A.4 Compass Observation Model

The compass measures the orientation of the vehicle directly. Therefore,

$$\mathbf{h}_\theta \left[\mathbf{x}_v\left(k\right), \mathbf{w}\left(k\right)\right] = \begin{bmatrix} \theta_v(k) + w_\theta(k) \end{bmatrix}$$

where $w_\theta(k)$ is assumed to be additive with standard deviation $\sigma_{\theta\theta} = 2°$.

Appendix B

The Mathematics of Modelling Errors

This appendix summarises the effect of modelling errors. It is a general presentation which allows for the case in which the state space of the true system differs from that of the system utilised in the filter. This generality is exploited in Section **??** which consider the (näive) use of a compass. The material in this section was derived in [11] and presented in [14].

This section generalises other modeling error results (e.g., Theorem 7.6 on page 248 of [10]) by considering nonlinear systems and assuming that the state space used in the filter is different from that used by the true system.

B.0.1 Modelling Error Framework

Suppose the "real world" system evolves according to the equations

$$\begin{aligned}\mathbf{x}_T(k) &= \mathbf{f}_T\left[\mathbf{x}_T(k-1), \mathbf{u}_T\left(k-1\right), \mathbf{v}_T\left(k\right)\right], & \text{(B.1)}\\ \mathbf{z}\left(k\right) &= \mathbf{h}_T\left[\mathbf{x}_T(k), \mathbf{u}_T\left(k\right), \mathbf{w}_T\left(k\right), k\right], & \text{(B.2)}\end{aligned}$$

where the subscript T denotes that these are the *true* equations of motion describing the behaviour of the system. The Vehicle Navigation System (VNS) employs a simplified or *approximate system* description of the true system model. There are many reasons for this, including lack of knowledge of the true system and the need to use a system of sufficiently small dimensions that it can be implemented in real-time. It is assumed that the VNS state, $\mathbf{x}\left(k\right)$, is related to the true system state according to the (unknown) transformation

$$\mathbf{x}\left(k\right) = \mathbf{g}\left[\mathbf{x}_T\left(k\right)\right]. \tag{B.3}$$

To simplify and focus subsequent sections, it is assumed that there is no error in the observation model and thus,

$$\mathbf{h}_T\left[\mathbf{x}_T(k), \mathbf{u}_T\left(k\right), \mathbf{w}_T\left(k\right), k\right] = \mathbf{h}\left[\mathbf{g}\left[\mathbf{x}_T(k)\right], \mathbf{u}\left(k\right), \mathbf{w}_T\left(k\right), k\right].$$

B.0.2 The Effects of Modelling Errors

Errors in the process noise must be compensated for by the well-known procedure of "tuning" (or enlarging) $\mathbf{Q}\left(k\right)$, the process noise covariance matrix, until the filter estimates are consistent [20]. Implicitly, this tuning procedure assumes that the residual unmodelled dynamics are stationary and temporally uncorrelated. However, this assumption is incorrect (the residual dynamics are temporally correlated) and this subsection derives an expression which shows how large the additional process noise must be.

Consider the prediction error introduced between time steps $k-1$ and k. Assuming the process noise is additive[1], the prediction error is

$$\begin{aligned}\tilde{\mathbf{x}}\left(k \mid k-1\right) &= \mathbf{g}\left[\mathbf{x}_T\left(k\right)\right] - \hat{\mathbf{x}}\left(k \mid k-1\right)\\ &= \mathbf{g}\left[\mathbf{f}_T\left[\mathbf{x}_T(k-1), \mathbf{u}_T\left(k-1\right), k-1\right] + \mathbf{v}_T\left(k\right)\right]\\ &\qquad - \mathbf{f}\left[\hat{\mathbf{x}}\left(k-1 \mid k-1\right), \mathbf{u}\left(k-1\right), k-1\right].\end{aligned} \tag{B.4}$$

[1]An additive form of the process model can be obtained by linearising the state transition equation. However, this introduces linearisation errors which have been discussed elsewhere [15].

The errors can be classified into three types: previous estimation errors ($\tilde{\mathbf{x}}\left(k-1 \mid k-1\right)$), process noise ($\mathbf{v}_T\left(k\right)$) and modelling errors. Linearising the first term on the right hand side of Equation B.4,

$$\begin{aligned}\mathbf{g}\left[\mathbf{x}_T\left(k\right)\right] \approx \mathbf{g}\left[\mathbf{f}_T\left[\mathbf{x}_T(k-1), \mathbf{u}_T\left(k-1\right), k-1\right]\right] \\ + \nabla\mathbf{g}\,\mathbf{v}_T\left(k-1\right).\end{aligned} \tag{B.5}$$

Let

$$\begin{aligned}\mathbf{e}\left[\mathbf{x}_T(k-1)\right] =& \mathbf{g}\left[\mathbf{f}_T\left[\mathbf{x}_T(k-1), \mathbf{u}_T\left(k-1\right), k-1\right]\right] \\ &- \mathbf{f}\left[\mathbf{g}\left[\mathbf{x}_T(k-1)\right], \mathbf{u}\left(k-1\right), k-1\right].\end{aligned} \tag{B.6}$$

This is the prediction error arising purely from the use of an incorrect process model. Noting that $\mathbf{g}\left[\mathbf{x}_T(k-1)\right] = \hat{\mathbf{x}}\left(k-1 \mid k-1\right) + \tilde{\mathbf{x}}\left(k-1 \mid k-1\right)$ and linearising about $\hat{\mathbf{x}}\left(k-1 \mid k-1\right)$, it can be shown that

$$\begin{aligned}&\mathbf{f}\left[\mathbf{g}\left[\mathbf{x}_T(k-1)\right], \mathbf{u}\left(k-1\right), k-1\right] \\ &\quad\approx \mathbf{f}\left[\hat{\mathbf{x}}\left(k-1 \mid k-1\right), \mathbf{u}\left(k-1\right), k-1\right] \\ &\quad+ \nabla\mathbf{f}\,\tilde{\mathbf{x}}\left(k-1 \mid k-1\right).\end{aligned} \tag{B.7}$$

Substituting Equations B.5, B.6, B.7 and **??** into Equation B.4, the prediction error becomes

$$\begin{aligned}\tilde{\mathbf{x}}\left(k \mid k-1\right) = \nabla\mathbf{f}\,\tilde{\mathbf{x}}\left(k-1 \mid k-1\right) \\ + \nabla\mathbf{g}\,\mathbf{v}_T\left(k-1\right) + \mathbf{e}\left[\mathbf{x}_T\left(k-1\right)\right].\end{aligned} \tag{B.8}$$

The first two terms account for the contributions from $\tilde{\mathbf{x}}\left(k-1 \mid k-1\right)$ and $\mathbf{v}_T\left(k\right)$. The third term is the contribution from the modeling error. It is a function of the true system, the approximate system model, and the structural relationship between the two.

The prediction covariance is found by taking the outer product and expectations of Equation B.8:

$$\begin{aligned}\mathbf{P}\left(k \mid k-1\right) = &\nabla\mathbf{f}\,\mathbf{P}\left(k-1 \mid k-1\right)\nabla^T\mathbf{f} \\ &+ \nabla\mathbf{g}\,\mathbf{Q}_T\left(k-1\right)\nabla^T\mathbf{g} + \mathbf{P}_{ee}\left(k-1\right) \\ &+ \nabla\mathbf{f}\,\mathbf{P}_{xe}\left(k-1 \mid k-1\right) \\ &+ \mathbf{P}_{xe}^T\left(k-1 \mid k-1\right)\nabla^T\mathbf{f}.\end{aligned} \tag{B.9}$$

The first two terms in this expression are the conventional linearised covariance prediction expressions. The last three terms are due to modeling errors. These require the propagation of two extra covariance terms,

$$\begin{aligned}\mathbf{P}_{ee}\left(k\right) &\triangleq \mathrm{E}\left[\mathbf{e}\left[\mathbf{x}_T(k)\right]\mathbf{e}^T\left[\mathbf{x}_T(k)\right]|\mathbf{Z}^k\right], \\ \mathbf{P}_{xe}\left(k-1 \mid k-1\right) &\triangleq \mathrm{E}\left[\tilde{\mathbf{x}}\left(k-1 \mid k-1\right)\mathbf{e}^T\left[\mathbf{x}_T(k-1)\right]|\mathbf{Z}^k\right].\end{aligned}$$

$\mathbf{P}_{ee}\left(k\right)$ is the mean squared value of the modelling error. $\mathbf{P}_{xe}\left(k-1 \mid k-1\right)$ accounts for the correlation which is introduced between $\tilde{\mathbf{x}}\left(k-1 \mid k-1\right)$ and the modeling error introduced when predicting from $k-1$ to k. Because $\mathbf{P}_{xe}\left(k-1 \mid k-1\right)$ is a function of the estimate, it has its own prediction-update structure. Substituting from Equation B.8 and taking expectations,

$$\begin{aligned}\mathbf{P}_{xe}\left(k \mid k-1\right) &= \mathrm{E}\left[\tilde{\mathbf{x}}\left(k \mid k-1\right)\mathbf{e}^T\left[\mathbf{x}_T(k)\right]|\mathbf{Z}^k\right] \\ &= \nabla\mathbf{g}\,\mathbf{P}_{xe}\left(k-1 \mid k-1\right) + \mathbf{P}_{ee}\left(k-1\right) \\ &\quad+ \mathrm{E}\left[\tilde{\mathbf{x}}\left(k-1 \mid k-1\right)\Delta\mathbf{e}^T\left[\mathbf{x}_T(k)\right]\right] \\ &\quad+ \mathrm{E}\left[\mathbf{e}\left[\mathbf{x}_T(k-1)\right]\Delta\mathbf{e}^T\left[\mathbf{x}_T(k)\right]\right]\end{aligned}$$

where $\Delta\mathbf{e}\left[\mathbf{x}_T(k)\right] = \mathbf{e}\left[\mathbf{x}_T(k)\right] - \mathbf{e}\left[\mathbf{x}_T(k-1)\right]$.

Taking the Taylor Series expansion of Equation **??**, the error in the estimate is

$$\tilde{\mathbf{x}}\left(k \mid k\right) = \left(\mathbf{I} - \mathbf{W}\left(k\right)\nabla_x\mathbf{h}\right)\tilde{\mathbf{x}}\left(k \mid k-1\right) + \mathbf{W}\left(k\right)\mathbf{w}\left(k\right)$$

Therefore,

$$\begin{aligned}\mathbf{P}_{xe}\left(k \mid k\right) &= \mathrm{E}\left[\tilde{\mathbf{x}}\left(k \mid k\right)\mathbf{e}^{T}\left[\mathbf{x}_{T}(k)\right]|\mathbf{Z}^{k}\right] \\ &= \left(\mathbf{I} - \mathbf{W}\left(k\right)\boldsymbol{\nabla}_{x}\mathbf{h}\right)\mathbf{P}_{xe}\left(k \mid k-1\right)\end{aligned}$$

By comparing Equation B.9 and the prediction covariance, $\mathbf{Q}\left(k\right)$ must satisfy the inequality

$$\begin{aligned}\mathbf{Q}\left(k\right) \geq \boldsymbol{\nabla}\mathbf{g}\,\mathbf{Q}_{T}\left(k\right)\boldsymbol{\nabla}^{T}\mathbf{g} &+ \boldsymbol{\nabla}\mathbf{f}\,\mathbf{P}_{xe}\left(k-1 \mid k-1\right) \\ &+ \mathbf{P}_{xe}^{T}\left(k-1 \mid k-1\right)\boldsymbol{\nabla}^{T}\mathbf{f} + \mathbf{P}_{ee}\left(k-1\right)\end{aligned} \tag{B.10}$$

for the prediction to be consistent. This demonstrates that it is not necessary to know the precise relationship between the true and modeled systems. However, even a relatively small error in a process model can lead to significant change in the value of the stabilising process noise[2].

This analysis reveals a number of important consequences induced by using inaccurate process models. First, the value of the modeling error terms will, in general, be time varying because the dynamics of the true and approximate systems are different. Therefore, an approximate system might model some behaviours of the true system accurately and other behaviours of the true system poorly. Second, the choice of the process model and sensor suite are coupled. Modeling errors lead to biases which, through Equation B.9, affect the level of process noise. The update tends to reduce these biases, but the magnitude of this reduction is determined by the value of gain $\mathbf{W}\left(k\right)$. Finally, a component of the error ($\mathbf{P}_{ee}\left(k\right)$), is a function only of $\mathbf{x}_{T}(k)$, and thus no choice of sensor suite will reduce or eliminate this term. Therefore, in certain circumstances the modelling errors can be such that it is *impossible* to develop a consistent filter for a finite value of $\mathbf{Q}\left(k\right)$.

[2]The analysis in this section assumes that the process noise is independent, leading to an equation similar in form to Equation **??**. However, the fundamental difficulty with modeling errors is that they introduce time correlated error terms and it is possible that improvements in performance can be met by removing the reliance on this independence assumption. One such algorithm is the Covariance Addition algorithm (CA), the dual to the Covariance Intersection algorithm, which finds the minimum consistent estimate of the sum of two random variables with unknown covariances [13]. Nicholson [22] has shown that CA yields more accurate and more robust results than "hand tuning" a Kalman filter over a set of standard simulation runs.

Bibliography

[1] Juan Andrade-Cetto, Teresa Vidal-Calleja, and Alberto Sanfeliu. Unscented transformation of vehicle states in slam. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA '05)*, pages 324–329, Barcelona, Spain, 18–20 April 2005. Available from: `http://www-iri.upc.es/english/conspubli.php?tipus=tots&autor%5B0%5D=13&autor%5B1%5D=31&autor%5B2%5D=66&autor%5B3%5D=70&autor%5B4%5D=159&autor%5B5%5D=47&fes=si` [cited August 13, 2006].

[2] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, New York NY, USA, 1988.

[3] J. A. Castellanos, J. M. M. Montiel, J. Neira and J. D. Tardós. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–952, October 1999.

[4] M. Csorba. *Simultaneous Localisation and Map Building*. PhD thesis, University of Oxford, 1997.

[5] M. Csorba and H. F. Durrant-Whyte. New Approach to Map Building Using Relative Position Estimates. In *Proceedings of the SPIE AeroSense Conference*, volume 3087, pages 115–125, Orlando, FL, USA, April 1997.

[6] M.W.M.G Dissanayake, H. Durrant-Whyte and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) Problem. In *Proceedings of the 6th International Symposium on Experimental Robotics*, Sydney, Australia, 1999.

[7] H. F. Durrant-Whyte. An autonomous guided vehicle for cargo handling applications. *International Journal of Robotics Research*, 15(5):407–440, October 1996.

[8] D. Gebre-Egziabher, G. H. Elkaim, J. David Powell and B. W. Parkinson. A non-linear, two-step estimation algorithm for calibrating solid-state strapdown magnetometers. In *Proceedings of the International Conference on Integrated Navigation Systems*, St. Petersburg, Russia, May 2001.

[9] P. W. Gibbens, G. Dissanayake and H. F. Durrant-Whyte. A Closed Form Solution to the Single Degree of Freedom Simultaneous Localisation and Map Building (SLAM) Problem. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, 12–15 December 2000. IEEE, IEEE.

[10] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, San Diego, CA, 1970.

[11] S. J. Julier. *Comprehensive Process Models for High-Speed Navigation*. PhD thesis, University of Oxford, October 1997.

[12] S. J. Julier and J. K. Uhlmann. A Counter Example to the Theory of Simultaneous Localization and Map Building. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, 21–26 May 2001.

[13] S. J. Julier and J. K. Uhlmann. General Decentralized Data Fusion With Covariance Intersection (CI). In D. Hall and J. Llinas, editor, *Handbook of Data Fusion*. CRC Press, Boca Raton FL, USA, 2001.

[14] S. J. Julier, J. K. Uhlmann and H. F. Durrant-Whyte. A new approach for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, March 2000.

[15] S. J. Julier, J. K. Uhlmann and H. F. Durrant-Whyte. A new approach for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, March 2000.

[16] Jonghyuk Kim and Salah Sukkarieh. Towards robust airborne slam in unknown wind environments. In *Proceedings of the International Conference on Robotics and Automiation (ICRA'06)*, Orlando, FL, USA, 15–19 May 2006. IEEE. Available from: `www.araa.asn.au/acra/acra2005/papers/kim_suhkarieh.pdf` [cited August 13, 2006].

[17] John J. Leonard and Richard J. Rikoski. Incorporation of delayed decision making into stochastic mapping. In *Experimental Robotics VII*, volume 271 of *Lecture Notes in Control and Information Sciences*, pages 533–542. Springer-Verlag, 2001. Available from: `http://citeseer.ist.psu.edu/leonard00incorporation.html` [cited August 13, 2006].

[18] J. J. Leonard and H. J. S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In D. Koditschek and J. Hollerbach, editors, *Ninth International Symposium on Robotics Research*, pages 1442–1447, Snowbird, Utah, 1999.

[19] Ruben Martinez-Cantin and Jose A. Castellanos. Unscented slam for large-scale outdoor environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005)*, pages 328–333, Edmonton, Alberta, Canada, 2–6 August 2005. Available from: `http://citeseer.ist.psu.edu/martinez-cantin05unscented.html` [cited August 13, 2006].

[20] P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, New York, NY, 1979.

[21] P. Newman. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. PhD thesis, University of Sydney, 1999.

[22] D. Nicholson. Distributed data fusion: Bae systems final report. Technical report, BAE Systems, Sowerby, Bristol, UK, May 2001.

[23] Mark Pupilli and Andrew Calway. Real-time visual slam with resilience to erratic motion. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conferene (CVPR '06)*, New York, NY, USA, 2006. IEEE. Available from: `http://www.cs.bris.ac.uk/Publications/pub_by_author.jsp?id=9819557` [cited August 13, 2006].

[24] S. F. Schmidt. Applications of state space methods to navigation problems. In C. T. Leondes, editor, *Advanced Control Systems*, volume 3, pages 293–340. Academic Press, 1966.

[25] R. Smith, M. Self and P. Cheeseman. *Estimating Uncertain Spatial Relationships in Robotics, Autonomous Robot Vehicles*. Springer-Verlag, 1989.