Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 – Nov. 03, 2001

Towards Constant Time SLAM using Postponement

Joss Knight, Andrew Davison and Ian Reid *
Department of Engineering Science
University of Oxford
Oxford OX1 3PJ, UK
[joss,ajd,ian]@robots.ox.ac.uk

Abstract

Many recent approaches to Simultaneous Localisation and Mapping (SLAM) use an Extended Kalman Filter (EKF) to update and maintain a map of vehicle location and multiple feature positions as a sensor moves through a scene. Although it is a highly powerful and well-used tool, it suffers from a well-known complexity problem, that the amount of computation at each recursion step is proportional to the square of the number of features in the map. In this paper we outline the Postponement technique which allows for much greater flexibility about when to use available processing time, while in no way affecting the optimality of the filter. It works by updating a constant-sized data set based on current measurements, which can be used to effect updates on all unobserved parts of the map at a later stage. By expanding the set of updated features as each new feature is observed we show that the full map update can be postponed indefinitely. We also demonstrate how Postponement can be used to improve the performance of sub-optimal algorithms by applying it to a simple constant time method.

1 Introduction

If the aim is to produce the most accurate map of a scene from a set of data collected by the sensors of a robot or other moving body, the best solution is always batch post-processing of the data in a computationally costly minimisation. Such systems are common in machine vision structure-from-motion applications. Pollefeys, for instance, uses a typical system [12], while McLauchlan's Variable State Dimension Filter is an attempt to cross the divide between batch and recursive techniques [10]. However, in SLAM for mobile robot navigation, the time factor is crucially important, since a robot must respond continuously to the data obtained: the map must be built *sequentially*. More strictly, it is desirable that new data can be incorporated into the map within the *constant time* processing interval available between synchronous movement steps. This constant time constraint is problematic for the EKF, commonly applied to SLAM problems, since when implemented correctly as a single filter with full representation of the coupled uncertainty in a map, the computational expense increases with map size. To fit constant time constraints, the full, *optimal* EKF must be modified.

In practice any method which reduces the computational complexity of the EKF is potentially beneficial. The simplest do not affect the optimality of the filter. Davison, Durrant-Whyte, Dissanayake and others have all noted that many map features give redundant information and can be deleted with little effect on vehicle localisation accuracy [2–4], resulting in a smaller, more rapidly processed map. In addition, Newman's Geometric Projection Filter [11], uses rapidly built relative maps.

Sub-optimal methods can provide speedier filtering at the cost of accuracy, by neglecting some of the coupling in map estimates. Nebot *et al.* [6] do exactly this explicitly, giving a filter with complexity linear in map size. Castellanos [1] and Leonard [9] both use static submapping strategies in which the map is divided up into groups of features which can be processed separately. Leonard's method results in constant time filter complexity, as does that of Julier [7], who combines Kalman Filtering and a technique called Covariance Intersection so that all cross-correlation information can be neglected.

Much of the problem with sub-optimal methods is that there can be unexpected effects on the filter, which cannot be predicted theoretically. The methods are usually shown to be *conservative* (where the overall map uncertainty is not less than it should be), but

*This work is supported by the UK's Engineering and Physical Sciences Research Council through a studentship to JGHK.

0-7803-6612-3/01/$10.00©2001 IEEE

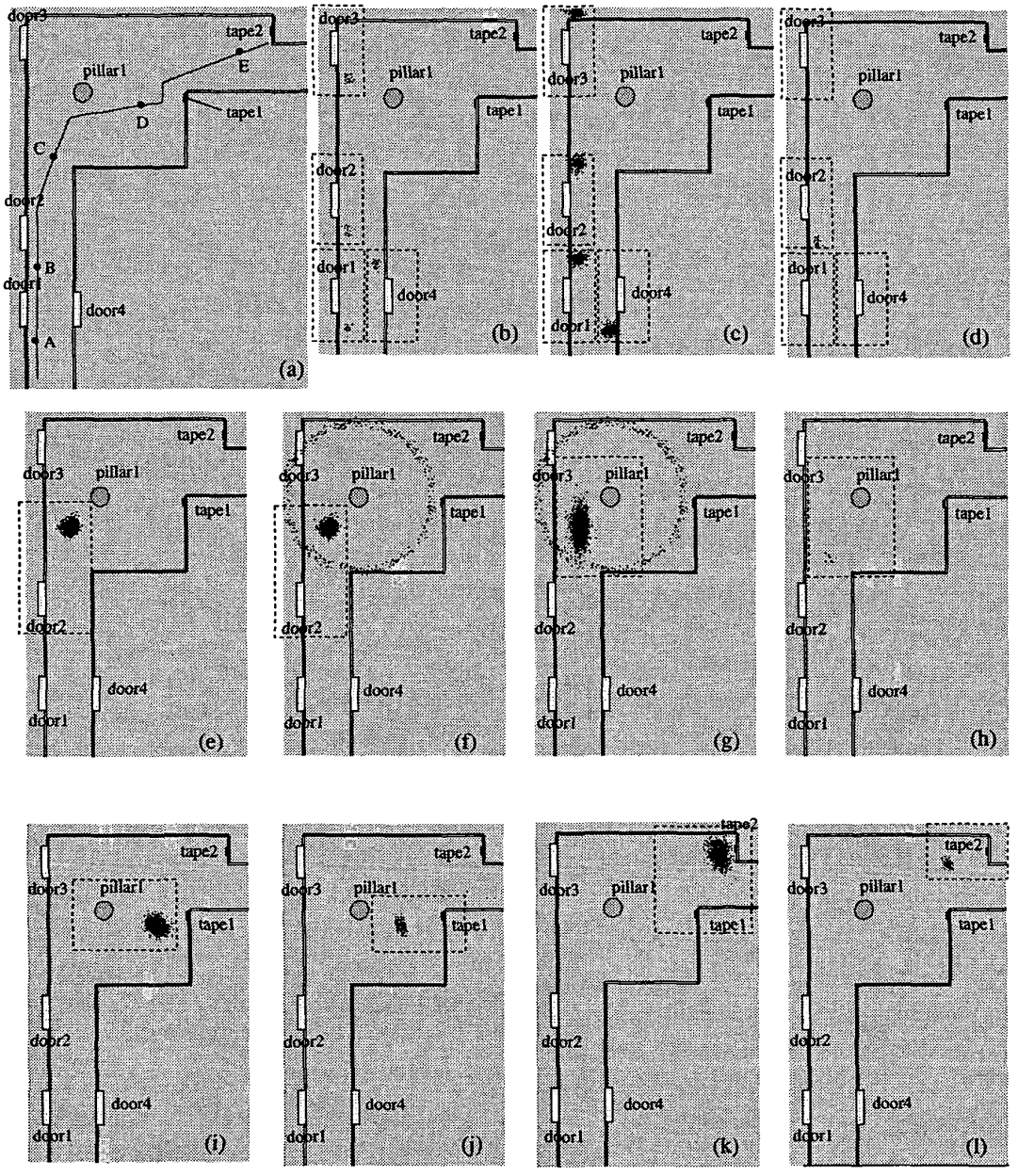


Figure 4: Experimental result: (a) An accurate map of the environment. The robot moves from A to E. The robot recognizes door1 at A, door2 at B, pillar1 at C, tape1 at D, and tape2 at E. (b) A roughly measured map. The dots are the initial robot-pose distribution based on the recognition of door1 at A. The rectangles of dotted lines represent base entities. (c) The robot pose after moving from A to B. (d) The robot pose fused by recognizing door2 at B. The base entity is door2. (e) The robot pose after moving from B to C. The base entity is door2. (f) The doughnut around pillar1 is the hypothesis generated by recognizing pillar1 at C. The robot pose is not overlapped with the doughnut and cannot be fused. (g) The robot pose after motion is now overlapped with the doughnut by diffusing the robot pose with the coordinate transformation from the door2 frame to the pillar1 frame. (h) The robot pose was fused. The base entity is pillar1. (i) The robot pose after moving from C to D. The base entity is pillar1. (j) The robot pose after recognizing tape1 at D. The base entity is tape1. (k) The robot pose moving from D to E. The base entity is tape1. (l) The robot pose after recognizing tape2 at E. The base entity is tape2.

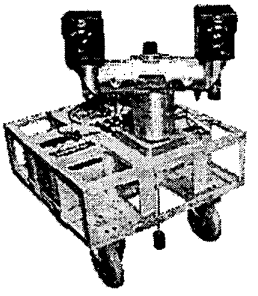


Figure 1: Head Yorick and vehicle GTI

this does not prove *consistency* (that the map will eventually converge to the true solution). A fairly comprehensive review of the various SLAM 'shortcut' techniques is given in [8], which includes a discussion of filter evaluation.

The filtering techniques described in this paper, while applicable to any EKF implementation of SLAM, have been devised with the vehicle and sensors of Figure 1 in mind. Fixated triangulation using a four-axis stereo head-eye platform provides the measurement data. While visual data provides a wealth of information for a variety of robotic tasks, not least greatly reducing the problems of data association, the tradeoff is a heavy cost in time (to find and fixate features) and a lack of depth accuracy (relative to sonar for example).

As a result of this, the original system as described in [2] takes care, in feature initialisation and selection, to generate a sparse map of high quality features. The next step, which this paper will describe, is Postponement, which provides an order of magnitude increase in the number of features that can be collected, by allowing most of the processing to be carried out at a later stage when more processing time may be available. Essentially, it allows us to gather up all the changes that would have been made to features that are not being observed into a set of constant size matrices, and alter these at each step instead until such time as a full map update is required. At the cost of occasional expensive updates that can be postponed indefinitely or carried out in background processing, most filter steps will be constant time. The final section shows Postponement working with partial decoupling, a constant time method which generates navigable maps the accuracy of which is greatly improved by the additional flexibility Postponement provides.

Since the original work on Postponement in [2], Guivant and Nebot have independently derived the 'compression' algorithm [5], which is effectively the same method except that the set of maintained features is defined by geographical boundaries rather than expanded dynamically (see §4.3).

2 Notation

The following outlines the notation used throughout the paper. Vectors will appear in bold type ($\mathbf{x}, \mathbf{y}$) and matrices in teletype ($\mathtt{M}, \mathtt{P}$). Map building notation will be as in [2]:

$\mathbf{x} = \begin{bmatrix} \mathbf{x}_v \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_i \\ \vdots \end{bmatrix}$ State vector consisting of vehicle state and feature states. The v subscript is dropped from $\mathbf{x}_v$ where it is not ambiguous.

$\mathtt{P} = \begin{bmatrix} \mathtt{P}_{\mathbf{xx}} & \mathtt{P}_{\mathbf{xy}_1} & \mathtt{P}_{\mathbf{xy}_2} & \cdots \\ \mathtt{P}_{\mathbf{y}_1\mathbf{x}} & \mathtt{P}_{\mathbf{y}_1\mathbf{y}_1} & \mathtt{P}_{\mathbf{y}_1\mathbf{y}_2} & \cdots \\ \mathtt{P}_{\mathbf{y}_2\mathbf{x}} & \mathtt{P}_{\mathbf{y}_2\mathbf{y}_1} & \mathtt{P}_{\mathbf{y}_2\mathbf{y}_2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$ Symmetric covariance matrix and its blocks. Note $\mathtt{P}_{\mathbf{y}_i\mathbf{y}_j} = \mathtt{P}_{\mathbf{y}_j\mathbf{y}_i}{}^{\top}$

In addition, $\mathbf{h}_i$ is the measurement equation for measurement of the ith feature.

3 EKF – The Full Filter

We begin by stating the well-known Extended Kalman Filter equations.

Prediction

Our state and covariance predictions at time $k+1$ are based on those at time k and use the state transition function $\mathbf{f}$ and its Jacobian, control inputs $\mathbf{u}$, and the process noise covariance $\mathtt{Q}$.

$$\begin{aligned} \mathbf{x}(k+1|k) &= \mathbf{f}(\mathbf{x}(k|k), \mathbf{u}(k)) \\ \mathtt{P}(k+1|k) &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(k|k)\mathtt{P}(k|k)\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\top}(k|k) + \mathtt{Q}(k) \end{aligned}$$

Update

The update to include recent measurements incorporates the innovation $\boldsymbol{\nu}$, which is the difference between the measurement and its prediction, the innovation covariance $\mathtt{S}$, and measurement noise $\mathtt{R}$.

$$\begin{aligned} \mathbf{x}(k+1|k+1) &= \mathbf{x}(k+1|k) + \mathtt{W}(k+1)\boldsymbol{\nu}(k+1) \\ \mathtt{P}(k+1|k+1) &= \mathtt{P}(k+1|k) - \mathtt{W}(k+1)\mathtt{S}(k+1)\mathtt{W}^{\top}(k+1) \end{aligned}$$

where

$$\mathtt{W}(k+1) = \mathtt{P}(k+1|k)\,\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\top}(k|k)\,\mathtt{S}^{-1}(k+1)$$

$$\mathtt{S}(k+1) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(k|k)\,\mathtt{P}(k+1|k)\,\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\top}(k|k) + \mathtt{R}(k+1)\ .$$

3.1 Implementation

The previous section states the known EKF equations. In many typical applications of the EKF the whole state vector is measured at every step, and the state transition function also affects the whole state, so the filter complexity is cubic in the state size. In SLAM, the state transition function affects only the vehicle state, and usually only a small subset of features are measured at each step. The resulting filter has complexity of order n^2, where n is the number of map features.

In this section we explain our rearrangement of the basic equations to give a separate equation for each feature state and covariance block. The k notation for time dependency is dropped since it is clear to which time step is referred. The initial derivation is based on the measurement of a single feature i at each time step.

3.1.1 Explanatory equations

First note the sparsity in the Jacobians,

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}_{\mathbf{v}}}{\partial \mathbf{x}_{\mathbf{v}}} & 0 & \cdots \\ 0 & \mathtt{I} & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix}, \quad \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} & 0 \cdots & \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} & 0 \cdots \end{bmatrix}$$

where here there has been a measurement of feature i. Also sparse is the process noise $\mathtt{Q} = [\mathtt{Q}_{\mathbf{v}}^{\top}\ \ 0\ \cdots]^{\top}$.

Multiplying out the equation for the innovation covariance S, it can now be seen that it is a constant time calculation (independent of map size):

$$\begin{aligned} \mathtt{S} = {} & \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} \mathtt{P}_{\mathbf{x}\mathbf{x}} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}}^{\top} + \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \mathtt{P}_{\mathbf{y}_i\mathbf{x}} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}}^{\top} \\ & + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} \mathtt{P}_{\mathbf{x}\mathbf{y}_i} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^{\top} + \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \mathtt{P}_{\mathbf{y}_i\mathbf{y}_i} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^{\top} + \mathtt{R}\ . \end{aligned} \tag{1}$$

One final expansion, of the gain term $\mathtt{WSW}^{\top}$, should make it clear how the summary equations of the next section come about:

$$\begin{aligned} \mathtt{WSW}^{\top} = {} & \begin{bmatrix} \mathtt{P}_{\mathbf{x}\mathbf{x}} \\ \mathtt{P}_{\mathbf{y}_1\mathbf{x}} \\ \mathtt{P}_{\mathbf{y}_2\mathbf{x}} \\ \vdots \end{bmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}}^{\top} \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} \begin{bmatrix} \mathtt{P}_{\mathbf{x}\mathbf{x}} & \mathtt{P}_{\mathbf{x}\mathbf{y}_1} & \mathtt{P}_{\mathbf{x}\mathbf{y}_2} \cdots \end{bmatrix} \\ & + \begin{bmatrix} \mathtt{P}_{\mathbf{x}\mathbf{y}_i} \\ \mathtt{P}_{\mathbf{y}_1\mathbf{y}_i} \\ \mathtt{P}_{\mathbf{y}_2\mathbf{y}_i} \\ \vdots \end{bmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^{\top} \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} \begin{bmatrix} \mathtt{P}_{\mathbf{x}\mathbf{x}} & \mathtt{P}_{\mathbf{x}\mathbf{y}_1} & \mathtt{P}_{\mathbf{x}\mathbf{y}_2} \cdots \end{bmatrix} \\ & + \begin{bmatrix} \mathtt{P}_{\mathbf{x}\mathbf{y}_i} \\ \mathtt{P}_{\mathbf{y}_1\mathbf{y}_i} \\ \mathtt{P}_{\mathbf{y}_2\mathbf{y}_i} \\ \vdots \end{bmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^{\top} \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \begin{bmatrix} \mathtt{P}_{\mathbf{y}_i\mathbf{x}} & \mathtt{P}_{\mathbf{y}_i\mathbf{y}_1} & \mathtt{P}_{\mathbf{y}_i\mathbf{y}_2} \cdots \end{bmatrix} \\ & + \begin{bmatrix} \mathtt{P}_{\mathbf{x}\mathbf{y}_i} \\ \mathtt{P}_{\mathbf{y}_1\mathbf{y}_i} \\ \mathtt{P}_{\mathbf{y}_2\mathbf{y}_i} \\ \vdots \end{bmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^{\top} \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \begin{bmatrix} \mathtt{P}_{\mathbf{y}_i\mathbf{x}} & \mathtt{P}_{\mathbf{y}_i\mathbf{y}_1} & \mathtt{P}_{\mathbf{y}_i\mathbf{y}_2} \cdots \end{bmatrix} \end{aligned}$$

noting that S is symmetric so $\mathtt{S}^{-\top} = \mathtt{S}^{-1}$.

3.1.2 Reformulation

First, define some intermediate vectors and matrices as follows:

$$\mathbf{m} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}}^{\top} \mathtt{S}^{-1} \boldsymbol{\nu} \qquad \mathbf{n} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^{\top} \mathtt{S}^{-1} \boldsymbol{\nu} \tag{2}$$

$$\mathtt{A} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}}^{\top} \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} \qquad \mathtt{B} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}}^{\top} \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \tag{3}$$

$$\mathtt{C} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^{\top} \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} = \mathtt{B}^{\top} \qquad \mathtt{D} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^{\top} \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \tag{4}$$

From these intermediates and the equations of the previous sections, the filter equations of Figure 2 can be derived. They show how to process each block of the state vector or covariance matrix separately.

3.1.3 Submaps

In the Postponement section we will wish to consider $\mathbf{y}_i$ as referring not to a single feature, but a group of features, or *submap feature*. To emphasise the distinction here we will call the submap $\mathbf{y}_I$. Let us now write the innovation covariance, as we may, in terms of Jacobians involving $\mathbf{y}_I$ and the single measured feature in the submap, i:

$$\begin{aligned} \mathtt{S} = {} & \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} \mathtt{P}_{\mathbf{x}\mathbf{x}} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}}^{\top} + \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_I} \mathtt{P}_{\mathbf{y}_I\mathbf{x}} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}}^{\top} \\ & + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{\mathbf{v}}} \mathtt{P}_{\mathbf{x}\mathbf{y}_I} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_I}^{\top} + \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_I} \mathtt{P}_{\mathbf{y}_I\mathbf{y}_I} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_I}^{\top} + \mathtt{R}\ . \end{aligned}$$

Since $\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_I} = [0 \cdots \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} 0 \cdots]$, it should be clear that this equation reduces back to the original equation (1). Similarly, replacing $\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}$ with $\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_I}$ in the expansion of $\mathtt{WSW}^{\top}$ is valid.

In fact, reassessing all the working in this section replacing $\mathbf{y}_i$ with $\mathbf{y}_I$ is perfectly valid. So, for instance,

Prediction – equations act on old values

$$\mathbf{x_v}(\text{pred}) = \mathbf{f_v}(\mathbf{x_v}, \mathbf{u})$$
$$\mathbf{y}_i(\text{pred}) = \mathbf{y}_i$$
$$\mathbf{y}_j(\text{pred}) = \mathbf{y}_j$$
$$\mathtt{P_{xx}}(\text{pred}) = \frac{\partial \mathbf{f_v}}{\partial \mathbf{x_v}} \mathtt{P_{xx}} \frac{\partial \mathbf{f_v}}{\partial \mathbf{x_v}}^{\top} + \mathtt{Q_v}$$
$$\mathtt{P_{xy}}_i(\text{pred}) = \frac{\partial \mathbf{f_v}}{\partial \mathbf{x_v}} \mathtt{P_{xy}}_i$$
$$\mathtt{P_{y}}_i\mathtt{y}_i(\text{pred}) = \mathtt{P_{y}}_i\mathtt{y}_i$$
$$\mathtt{P_{xy}}_j(\text{pred}) = \frac{\partial \mathbf{f_v}}{\partial \mathbf{x_v}} \mathtt{P_{xy}}_j$$
$$\mathtt{P_{y}}_i\mathtt{y}_j(\text{pred}) = \mathtt{P_{y}}_i\mathtt{y}_j$$
$$\mathtt{P_{y}}_j\mathtt{y}_k(\text{pred}) = \mathtt{P_{y}}_j\mathtt{y}_k$$

Update – equations act on predicted values

$$\mathbf{x_v}(\text{new}) = \mathbf{x_v} + \mathtt{P_{xx}}\mathbf{m} + \mathtt{P_{xy}}_i\mathbf{n}$$
$$\mathbf{y}_i(\text{new}) = \mathbf{y}_i + \mathtt{P_{y}}_i\mathtt{x}\mathbf{m} + \mathtt{P_{y}}_i\mathtt{y}_i\mathbf{n}$$
$$\mathbf{y}_j(\text{new}) = \mathbf{y}_j + \mathtt{P_{y}}_j\mathtt{x}\mathbf{m} + \mathtt{P_{y}}_j\mathtt{y}_i\mathbf{n}$$
$$\mathtt{P_{xx}}(\text{new}) = \mathtt{P_{xx}} - (\mathtt{P_{xx}AP_{xx}} + \mathtt{P_{xx}BP_{y}}_i\mathtt{x} + \mathtt{P_{xy}}_i\mathtt{CP_{xx}} + \mathtt{P_{xy}}_i\mathtt{DP_{y}}_i\mathtt{x})$$
$$\mathtt{P_{xy}}_i(\text{new}) = \mathtt{P_{xy}}_i - (\mathtt{P_{xx}AP_{xy}}_i + \mathtt{P_{xx}BP_{y}}_i\mathtt{y}_i + \mathtt{P_{xy}}_i\mathtt{CP_{xy}}_i + \mathtt{P_{xy}}_i\mathtt{DP_{y}}_i\mathtt{y}_i)$$
$$\mathtt{P_{y}}_i\mathtt{y}_i(\text{new}) = \mathtt{P_{y}}_i\mathtt{y}_i - (\mathtt{P_{y}}_i\mathtt{xAP_{xy}}_i + \mathtt{P_{y}}_i\mathtt{xBP_{y}}_i\mathtt{y}_i + \mathtt{P_{y}}_i\mathtt{y}_i\mathtt{CP_{xy}}_i + \mathtt{P_{y}}_i\mathtt{y}_i\mathtt{DP_{y}}_i\mathtt{y}_i)$$
$$\mathtt{P_{xy}}_j(\text{new}) = \mathtt{P_{xy}}_j - (\mathtt{P_{xx}AP_{xy}}_j + \mathtt{P_{xx}BP_{y}}_i\mathtt{y}_j + \mathtt{P_{xy}}_i\mathtt{CP_{xy}}_j + \mathtt{P_{xy}}_i\mathtt{DP_{y}}_i\mathtt{y}_j) \quad (5)$$
$$\mathtt{P_{y}}_i\mathtt{y}_j(\text{new}) = \mathtt{P_{y}}_i\mathtt{y}_j - (\mathtt{P_{y}}_i\mathtt{xAP_{xy}}_j + \mathtt{P_{y}}_i\mathtt{xBP_{y}}_i\mathtt{y}_j + \mathtt{P_{y}}_i\mathtt{y}_i\mathtt{CP_{xy}}_j + \mathtt{P_{y}}_i\mathtt{y}_i\mathtt{DP_{y}}_i\mathtt{y}_j) \quad (6)$$
$$\mathtt{P_{y}}_j\mathtt{y}_k(\text{new}) = \mathtt{P_{y}}_j\mathtt{y}_k - (\mathtt{P_{y}}_j\mathtt{xAP_{xy}}_k + \mathtt{P_{y}}_j\mathtt{xBP_{y}}_i\mathtt{y}_k + \mathtt{P_{y}}_j\mathtt{y}_i\mathtt{CP_{xy}}_k + \mathtt{P_{y}}_j\mathtt{y}_i\mathtt{DP_{y}}_i\mathtt{y}_k) \quad (7)$$

Figure 2: A summary of the reformulated EKF equations. Subscript i refers to the observed feature (or submap), j and k refer to other features. Thus the equations for $\mathtt{P_{y}}_j\mathtt{y}_k$ for instance work for any cross-covariance between two unobserved features, including if $j = k$.

equation (6) for the update of $\mathtt{P_{y}}_i\mathtt{y}_j$ in Figure 2 now affects the matrix block $\mathtt{P_{y}}_I\mathtt{y}_j$, which is a tall matrix containing all the cross-covariances between features in the submap and feature j. Some of the intermediate vectors and matrices ($\mathbf{n}$, $\mathtt{B}$, $\mathtt{C}$, and $\mathtt{D}$) will expand in size to take account of this.

In [2] Postponement is derived without submaps in mind, and to maintain consistency with this the $\mathbf{y}_I$ notation is now dropped. However, note that in future derivation, wherever $\mathbf{y}_i$ is used, it may be replaced with $\mathbf{y}_I$ and thereby represent a submap instead. It should also be noted that i may easily represent multiple features measured simultaneously, although, depending on the implementation, it may be easier instead to present each measurement to the filter separately, which will give the same results.

4 Postponement

In the last section we derived an order n^2 filter with a separate equation for each map element and covariance block. The problem is that measurement of any one feature will affect every part of the state and covariance. In this section we illustrate how the changes that are occurring to data not directly related to the measured feature can be accumulated in a set of constant sized vectors and matrices, and the filter steps replaced with constant time recursive updates of this *postponement data*. The only ordinary updates carried out affect a constant sized portion of the state and covariance.

4.1 Derivation

The full derivation of Postponement is lengthy and the space is best used for presenting the full algorithm. Instead the reader is referred to [2], and only a representative part of the derivation is shown here. Let us rewrite equations (5) and (6), for the update of $\mathtt{P_{xy}}_j$ and $\mathtt{P_{y}}_i\mathtt{y}_j$ as

$$\mathtt{P_{xy}}_j(\text{new}) = \mathtt{EP_{xy}}_j + \mathtt{FP_{y}}_i\mathtt{y}_j$$
$$\mathtt{P_{y}}_i\mathtt{y}_j(\text{new}) = \mathtt{GP_{y}}_i\mathtt{y}_j + \mathtt{HP_{xy}}_j$$

where

$$\mathtt{E} = \mathtt{I} - [\mathtt{P_{xx}A} + \mathtt{P_{xy}}_i\mathtt{C}] \quad (8)$$
$$\mathtt{F} = -[\mathtt{P_{xx}B} + \mathtt{P_{xy}}_i\mathtt{D}] \quad (9)$$
$$\mathtt{G} = \mathtt{I} - [\mathtt{P_{y}}_i\mathtt{xB} + \mathtt{P_{y}}_i\mathtt{y}_i\mathtt{D}] \quad (10)$$
$$\mathtt{H} = -[\mathtt{P_{y}}_i\mathtt{xA} + \mathtt{P_{y}}_i\mathtt{y}_i\mathtt{C}] \ . \quad (11)$$

These equations are mutually recursive, that is, we can write the overall update of $\mathtt{P_{xy}}_j$ and $\mathtt{P_{y}}_i\mathtt{y}_j$ after k update steps in terms of recursively updated matrices $\mathtt{E_T}$, $\mathtt{F_T}$, $\mathtt{G_T}$ and $\mathtt{H_T}$ acting on the old values (referred to by (0)):

$$\mathtt{P_{xy}}_j(\text{new}) = \mathtt{E_T P_{xy}}_j(0) + \mathtt{F_T P_{y}}_i\mathtt{y}_j(0) \quad (12)$$
$$\mathtt{P_{y}}_i\mathtt{y}_j(\text{new}) = \mathtt{G_T P_{y}}_i\mathtt{y}_j(0) + \mathtt{H_T P_{xy}}_j(0) \quad (13)$$

where the recursive updates for $\mathtt{E_T}$ to $\mathtt{H_T}$ are

$$\mathtt{E_T}(\text{new}) = \mathtt{EE_T} + \mathtt{FH_T} \qquad \mathtt{F_T}(\text{new}) = \mathtt{EF_T} + \mathtt{FG_T}$$
$$\mathtt{G_T}(\text{new}) = \mathtt{GG_T} + \mathtt{HF_T} \qquad \mathtt{H_T}(\text{new}) = \mathtt{GH_T} + \mathtt{HE_T} \ .$$

This is true while observing any feature from the same submap i, and is a bounded amount of computation.

The insertion of equations (12) and (13) into (7) shows a similar argument for $\mathtt{P}_{\mathbf{y}_j\mathbf{y}_k}$, and the equations for $\mathbf{y}_j$ can be manipulated on similar grounds. The prediction step is easily incorporated.

4.2 Algorithm summary

Here we summarise the basic Postponement process. It uses ten matrices and vectors of postponement data, $\mathbf{m}_\mathtt{T}$, $\mathbf{n}_\mathtt{T}$, $\mathtt{A}_\mathtt{T}$, $\mathtt{B}_\mathtt{T}$, $\mathtt{C}_\mathtt{T}$, $\mathtt{D}_\mathtt{T}$, $\mathtt{E}_\mathtt{T}$, $\mathtt{F}_\mathtt{T}$, $\mathtt{G}_\mathtt{T}$ and $\mathtt{H}_\mathtt{T}$. The constant-sized set of state and covariance data always kept up to date, that is, $\mathbf{x}_\mathtt{v}$, $\mathbf{y}_i$, $\mathtt{P}_{\mathbf{x}\mathbf{x}}$, $\mathtt{P}_{\mathbf{x}\mathbf{y}_i}$ and $\mathtt{P}_{\mathbf{y}_i\mathbf{y}_i}$, are referred to as the *up-to-date data*.

1. The process begins when observations start to be made of a new feature or submap i. The state and covariance are currently fully up to date. Start by setting all the postponement data to zero, except $\mathtt{E}_\mathtt{T}$ and $\mathtt{G}_\mathtt{T}$ which are set to the identity.

2. **Prediction.** When the sensor moves, alter the up-to-date data, and change $\mathtt{E}_\mathtt{T}$ and $\mathtt{F}_\mathtt{T}$ as follows:

$$\begin{aligned}\mathtt{E}_\mathtt{T}(\text{pred}) &= \frac{\partial \mathbf{f}_\mathtt{v}}{\partial \mathbf{x}_\mathtt{v}}\mathtt{E}_\mathtt{T}\\ \mathtt{F}_\mathtt{T}(\text{pred}) &= \frac{\partial \mathbf{f}_\mathtt{v}}{\partial \mathbf{x}_\mathtt{v}}\mathtt{F}_\mathtt{T}\end{aligned}$$

3. **Update.** At every measurement of a feature in i, calculate vector $\mathbf{m}$ and $\mathbf{n}$ as in (2), $\mathtt{A}$ to $\mathtt{D}$ as in (3) and (4), and $\mathtt{E}$ to $\mathtt{H}$ as in (8) to (11).

   Now update the up-to-date data, as shown in Figure 2, and change the postponement data as follows:

$$\begin{aligned}\mathbf{m}_\mathtt{T}(\text{new}) &= \mathbf{m}_\mathtt{T} + \mathtt{E}_\mathtt{T}^\mathsf{T}\mathbf{m} + \mathtt{H}_\mathtt{T}^\mathsf{T}\mathbf{n}\\ \mathbf{n}_\mathtt{T}(\text{new}) &= \mathbf{n}_\mathtt{T} + \mathtt{F}_\mathtt{T}^\mathsf{T}\mathbf{m} + \mathtt{G}_\mathtt{T}^\mathsf{T}\mathbf{n}\\ \mathtt{A}_\mathtt{T}(\text{new}) &= \mathtt{A}_\mathtt{T} + \mathtt{E}_\mathtt{T}^\mathsf{T}\mathtt{A}\mathtt{E}_\mathtt{T} + \mathtt{E}_\mathtt{T}^\mathsf{T}\mathtt{B}\mathtt{H}_\mathtt{T} + \mathtt{H}_\mathtt{T}^\mathsf{T}\mathtt{C}\mathtt{E}_\mathtt{T} + \mathtt{H}_\mathtt{T}^\mathsf{T}\mathtt{D}\mathtt{H}_\mathtt{T}\\ \mathtt{B}_\mathtt{T}(\text{new}) &= \mathtt{B}_\mathtt{T} + \mathtt{E}_\mathtt{T}^\mathsf{T}\mathtt{A}\mathtt{F}_\mathtt{T} + \mathtt{E}_\mathtt{T}^\mathsf{T}\mathtt{B}\mathtt{G}_\mathtt{T} + \mathtt{H}_\mathtt{T}^\mathsf{T}\mathtt{C}\mathtt{F}_\mathtt{T} + \mathtt{H}_\mathtt{T}^\mathsf{T}\mathtt{D}\mathtt{G}_\mathtt{T}\\ \mathtt{C}_\mathtt{T}(\text{new}) &= \mathtt{C}_\mathtt{T} + \mathtt{F}_\mathtt{T}^\mathsf{T}\mathtt{A}\mathtt{E}_\mathtt{T} + \mathtt{F}_\mathtt{T}^\mathsf{T}\mathtt{B}\mathtt{H}_\mathtt{T} + \mathtt{G}_\mathtt{T}^\mathsf{T}\mathtt{C}\mathtt{E}_\mathtt{T} + \mathtt{G}_\mathtt{T}^\mathsf{T}\mathtt{D}\mathtt{H}_\mathtt{T}\\ \mathtt{D}_\mathtt{T}(\text{new}) &= \mathtt{D}_\mathtt{T} + \mathtt{F}_\mathtt{T}^\mathsf{T}\mathtt{A}\mathtt{F}_\mathtt{T} + \mathtt{F}_\mathtt{T}^\mathsf{T}\mathtt{B}\mathtt{G}_\mathtt{T} + \mathtt{G}_\mathtt{T}^\mathsf{T}\mathtt{C}\mathtt{F}_\mathtt{T} + \mathtt{G}_\mathtt{T}^\mathsf{T}\mathtt{D}\mathtt{G}_\mathtt{T}\\ \mathtt{E}_\mathtt{T}(\text{new}) &= \mathtt{E}\mathtt{E}_\mathtt{T} + \mathtt{F}\mathtt{H}_\mathtt{T}\\ \mathtt{F}_\mathtt{T}(\text{new}) &= \mathtt{E}\mathtt{F}_\mathtt{T} + \mathtt{F}\mathtt{G}_\mathtt{T}\\ \mathtt{G}_\mathtt{T}(\text{new}) &= \mathtt{G}\mathtt{G}_\mathtt{T} + \mathtt{H}\mathtt{F}_\mathtt{T}\\ \mathtt{H}_\mathtt{T}(\text{new}) &= \mathtt{G}\mathtt{H}_\mathtt{T} + \mathtt{H}\mathtt{E}_\mathtt{T}\end{aligned}$$

It remains to show how to carry out a full update of the state and covariance following a period of postponement (when a new submap is observed). Here (0) refers to the value at the last full update.

$$\begin{aligned}\mathbf{y}_j(\text{new}) &= \mathbf{y}_j(0) + \mathtt{P}_{\mathbf{y}_j\mathbf{x}}(0)\mathbf{m}_\mathtt{T} + \mathtt{P}_{\mathbf{y}_j\mathbf{y}_i}(0)\mathbf{n}_\mathtt{T}\\ \mathtt{P}_{\mathbf{y}_j\mathbf{y}_k}(\text{new}) &= \mathtt{P}_{\mathbf{y}_j\mathbf{y}_k}(0) -\\ &\quad (\mathtt{P}_{\mathbf{y}_j\mathbf{x}}(0)\mathtt{A}_\mathtt{T}\mathtt{P}_{\mathbf{x}\mathbf{y}_k}(0) + \mathtt{P}_{\mathbf{y}_j\mathbf{x}}(0)\mathtt{B}_\mathtt{T}\mathtt{P}_{\mathbf{y}_i\mathbf{y}_k}(0) +\\ &\quad \mathtt{P}_{\mathbf{y}_j\mathbf{y}_i}(0)\mathtt{C}_\mathtt{T}\mathtt{P}_{\mathbf{x}\mathbf{y}_k}(0) + \mathtt{P}_{\mathbf{y}_j\mathbf{y}_i}(0)\mathtt{D}_\mathtt{T}\mathtt{P}_{\mathbf{y}_i\mathbf{y}_k}(0))\\ \mathtt{P}_{\mathbf{x}\mathbf{y}_j}(\text{new}) &= \mathtt{E}_\mathtt{T}\mathtt{P}_{\mathbf{x}\mathbf{y}_j}(0) + \mathtt{F}_\mathtt{T}\mathtt{P}_{\mathbf{y}_i\mathbf{y}_j}(0)\\ \mathtt{P}_{\mathbf{y}_i\mathbf{y}_j}(\text{new}) &= \mathtt{G}_\mathtt{T}\mathtt{P}_{\mathbf{y}_i\mathbf{y}_j}(0) + \mathtt{H}_\mathtt{T}\mathtt{P}_{\mathbf{x}\mathbf{y}_j}(0)\end{aligned}$$

The computational complexity of this update increases linearly with submap size. So Postponement will cause an overall reduction in computation if submap features are measured multiple times (although some account must be taken of the time required to update the postponement data at each step). The saving could be very large, for instance if the vehicle explores the same submap for a long time.

4.3 Dynamic submap selection

Postponement as derived in [2] is intended to postpone full update while the same feature is being observed repeatedly. We have shown here how this extends easily to individual observations made within the same submap of features. This extension carries with it issues of how best to divide the map into submaps. One particular limitation of other submap-based methods is that they require the map space to be predivided geographically, with no guarantees that there will be a consistent number of features in each submap, or that the vehicle will spend any length of time there. Instead, we would like to consider our submap as being the set of features most recently observed, rather than defined by geographical boundaries. We can think of it as being a *cache* of features whose update is cheap, but which ultimately must be 'swapped out'. Naturally, an ideal cache would contain the set of features about to be observed as well as those just measured, which would result in the longest possible time before a non-cache feature was observed and a full map update was necessary.

Thankfully, forming such a feature set is quite simple. We can *grow* the postponement data to take account of a newly observed feature, expanding the submap size until processor time is available for a full map update or it reaches some size limit.

4.3.1 Growing the submap

Let us say we have a submap containing M features, and the sensor observes a postponed feature i' which is not currently in the submap and therefore out of date. First we use the current postponement matrices

to bring up to date this new feature's state $\mathbf{y}_{i'}$, and the covariance blocks $\mathbf{P}_{\mathbf{x}\mathbf{y}_{i'}}$, $\mathbf{P}_{\mathbf{y}_{i'}\mathbf{y}_{i'}}$, and $\mathbf{P}_{\mathbf{y}_i\mathbf{y}_{i'}}$, where the latter refers of course to all the cross-covariances between features in the submap and feature i'.

Now this feature can be included in the current submap as the $(M+1)$'th feature without any effect on the next full map update, if the postponement data are modified as follows:

$$\mathbf{m}_T(\text{new}) = \mathbf{m}_T \qquad \mathbf{n}_T(\text{new}) = \begin{bmatrix} \mathbf{n}_T \\ \mathbf{0} \end{bmatrix}$$

$$\mathtt{A}_T(\text{new}) = \mathtt{A}_T \qquad \mathtt{B}_T(\text{new}) = \begin{bmatrix} \mathtt{B}_T & 0 \end{bmatrix}$$

$$\mathtt{C}_T(\text{new}) = \begin{bmatrix} \mathtt{C}_T \\ 0 \end{bmatrix} \qquad \mathtt{D}_T(\text{new}) = \begin{bmatrix} \mathtt{D}_T & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathtt{E}_T(\text{new}) = \mathtt{E}_T \qquad \mathtt{F}_T(\text{new}) = \begin{bmatrix} \mathtt{F}_T & 0 \end{bmatrix}$$

$$\mathtt{G}_T(\text{new}) = \begin{bmatrix} \mathtt{G}_T & 0 \\ -(\mathbf{P}_{\mathbf{y}_{i'}\mathbf{x}}(0)\mathtt{B}_T + \mathbf{P}_{\mathbf{y}_{i'}\mathbf{y}_i}(0)\mathtt{D}_T) & \mathtt{I} \end{bmatrix}$$

$$\mathtt{H}_T(\text{new}) = \begin{bmatrix} \mathtt{H}_T \\ -(\mathbf{P}_{\mathbf{y}_{i'}\mathbf{x}}(0)\mathtt{A}_T + \mathbf{P}_{\mathbf{y}_{i'}\mathbf{y}_i}(0)\mathtt{C}_T) \end{bmatrix}$$

The size of the added blocks can be gleaned from the full update equations of §4.2. Derivation of the above is not given, but it should be clear that in the case of $\mathtt{G}_T$ and $\mathtt{H}_T$ the added blocks ensure correct update of $\mathbf{P}_{\mathbf{y}_{i'}\mathbf{y}_j}$, which is now part of $\mathbf{P}_{\mathbf{y}_i\mathbf{y}_j}$, and everywhere else the zero blocks simply remove it from the equations.

4.4 New features

The previous section describes how postponed features can be incorporated into the submap. Features which have never been observed before can be dealt with in a similar way. They must become part of the submap, and the postponement data expanded as detailed in the previous section, excepting $\mathtt{G}_T$ and $\mathtt{H}_T$. The cross-covariance of a new feature i with map feature j is initialised with $\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}\mathbf{P}_{\mathbf{x}\mathbf{y}_j}$. Therefore $\mathtt{G}_T$ and $\mathtt{H}_T$ are modified as follows:

$$\mathtt{G}_T(\text{new}) = \begin{bmatrix} \mathtt{G}_T & 0 \\ \frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}\mathtt{F}_T & 0 \end{bmatrix} \qquad \mathtt{H}_T(\text{new}) = \begin{bmatrix} \mathtt{H}_T \\ \frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}\mathtt{E}_T \end{bmatrix}$$

4.5 Results

It is not feasible with our hardware to carry out tests of our modified filter on large-scale scenes at this stage. However, Postponement is a simple reworking of the filter equations and synthetic scenes tested in simulation are sufficient to demonstrate the system's performance.

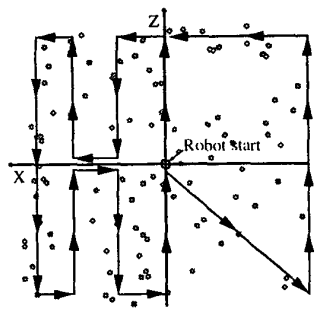
The filter was tested on multiple randomly generated maps of 100 features (Figure 3(a)). The simulated robot has identical parameters to that of the real system, and the system uses many of the same autonomous capabilities for choosing which feature to observe at any one time. This includes data association restrictions and simulated occlusion.

Given that every SLAM system has very different parameters, citing map accuracy data is somewhat meaningless. Suffice it to say that the results are the same as for the full filter, as required. To illustrate, Figure 3(b) shows true and estimated robot and feature positions after a single traverse of the map. Most telling, however, is Figure 3(c) which shows a close-up of the time spent filtering during a second map traverse. The full filter and Postponement filter were run in parallel. Note the sparsely occurring single step spikes, representing full map updates of the postponed map. Over 18 different maps the robot took 50 measurements per metre, and only 0.78% of time steps included full map updates. Overall, the Postponement filter ran in 37% of the full filter time.

4.6 Discussion

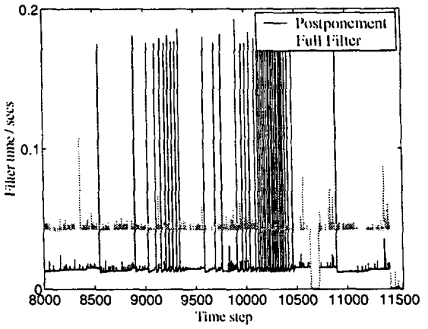
Although Postponement will normally conserve processing time, this is not its most useful attribute. The key feature of Postponement is that it gives the system considerable control over exactly when to use available processing time. The use of dynamic submap selection means that submap sizes do not need to be fixed, but can keep increasing until the time is available to carry out a full update. Many extensions to this technique are possible to achieve even faster execution. The only features that must be up to date are those being observed. When a new submap is formed the postponement data can be retained and used to update features before they are inserted into the submap.

Two other important points have been noted by Guivant and Nebot regarding their parallel compression algorithm [5]. Firstly, for reasons just mentioned, much of the full map update could be carried out as background processing, taking advantage of all available processing time, such as when the CPU is waiting on I/O. Secondly, postponement allows a far greater number of observations to be made of each feature with no computational impact on the full update, resulting in a considerably slower rate of increase in uncertainty as the vehicle explores the map. As a result, much larger exploration loops could be closed robustly.



(a) Plan view of a randomly generated scene of 100 features, and the path of the robot. The Scene is 20m×2m×20m, robot is approximately a cube of side 0.5m

(b) True (grey) and estimated (black) map after a single traverse, with 3σ error ellipses.



(c) Filter processing time chart comparing full filter to filter with Postponement and a submap size limit of 10. Simulation was run on a 600MHz Pentium III.

Figure 3: Results of Postponement tests

5 Partial Decoupling – an Application of Postponement

The previous section explains and evaluates the Postponement technique, showing how it provides considerable control over use of processing time. Fundamentally, however, the overall complexity of order n^2 is unchanged, and this could still prove a problem in very large maps.

The solution must be to use a sub-optimal approximation to the full EKF. There are many available, and most require the system to select a portion of the map for which some coupling information is discarded. Ideally, this is as small a portion as possible, and this is where Postponement can be beneficial. It enables us to process a much larger portion of the map optimally, since the dominant parameter for processing speed is the submap size, not the size of the set of features for which full coupling is retained.

To illustrate this we introduce a simple technique called partial decoupling. A portion of the map is kept up to date with the optimal filter, but the rest is decoupled from it and the vehicle. The algorithm is not consistent, but it can be used to produce distorted but navigable maps which do not diverge. As a general rule, maps must be navigable, but inaccuracy can be tolerated since it can always be corrected by off-line processing of sensor data. Partial decoupling simply provides a compromise between the processing expense of the optimal filter and the inaccuracy of a fully decoupled filter.

5.1 Definition

Partition the state and covariance as follows:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_A \\ \mathbf{x}_B \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{AA} & \mathbf{P}_{AB} \\ \mathbf{P}_{AB}^T & \mathbf{P}_{BB} \end{bmatrix}$$

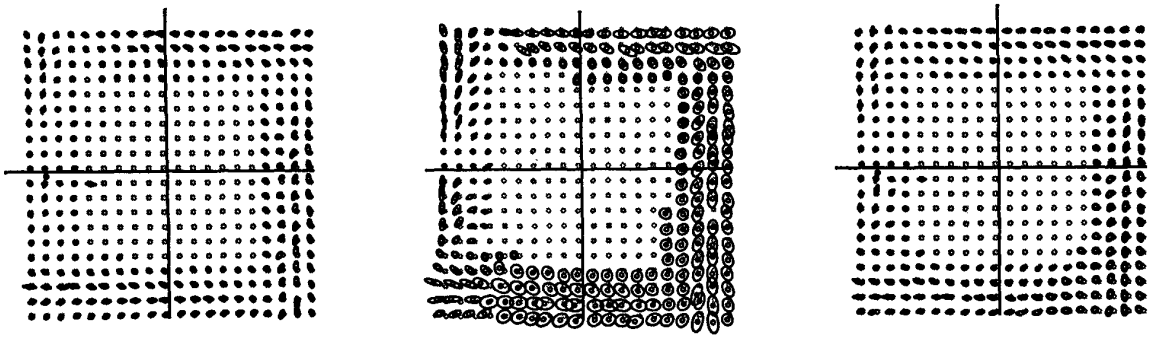
Partition A consists of the vehicle state and a fixed sized set of the most recently observed features, and therefore may change at every observation. $\mathbf{x}_A$ and $\mathbf{P}_{AA}$ are updated using the optimal algorithm. $\mathbf{x}_B$ and the covariances of partition B features are left unchanged, and the remaining cross-covariances in $\mathbf{P}$ are set to zero. The effect is to decouple features that have not been observed for a long time.

The technique is provably not conservative, but as might be expected, the severity of its inconsistency depends on the size of partition A, which is why Postponement can improve its performance.

5.2 Results and Discussion

To illustrate partial decoupling we used a 40m×40m map of 400 evenly spaced features (Figure 4). The vehicle travels around the edge of the map resulting in the longest possible time before it returns to the start. There will therefore likely be some discontinuity in the map when the vehicle adjusts its position when the loop is closed, which would be smoothed out in a working system.

Figure 4(b) shows the map generated for a partition A size of 40 features. The distortion and discontinuity are clearly apparent. In Figure 4(c) the partition A size was 80 features and the observation rate doubled.



(a) 40m×40m 'corridor' map generated in 1 circuit by the optimal filter. 264 features were observed.

(b) Corridor map generated in 1 circuit using Partial Decoupling with a partition A size of 40. 269 features were observed.

(c) Corridor map generated in 1 circuit using Partial Decoupling with a partition A size of 80. 268 features were observed.

Figure 4: Results of Partial Decoupling tests

However, Postponement was used with a submap size of 10, which enabled the vehicle to complete the circuit in the same time while producing an improved map.

Partial decoupling is flawed in that it is not conservative, yet Postponement can enable even this method to produce useful maps, demonstrating that it is a very effective tool in the effort to implement SLAM with no computational complexity issues.

6 Summary and Conclusions

In this paper we have derived the Postponement technique for aiding the computational efficiency of the Extended Kalman Filter used in SLAM. We have shown it to enable considerable control over the scheduling of processing time while retaining filter optimality, and we have demonstrated how it can be used to improve the performance of sub-optimal filter approximations.

References

[1] J. A. Castellanos, M. Devy, and J. D. Tardós. Simultaneous localisation and map building for mobile robots: A landmark-based approach. In *IEEE International Conference on Robotics and Automation Workshop on Mobile Robot Navigation and Mapping*, 2000.

[2] A. Davison. *Mobile Robot Navigation Using Active Vision.* PhD thesis, Robotics Research Group, Oxford University Department of Engineering Science, Feb. 1998. Full text available at `www.robots.ox.ac.uk/~ajd/`.

[3] G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *IEEE International Conference on Robotics and Automation*, San Francisco, USA, Apr. 2000.

[4] H. F. Durrant-Whyte, M. W. M. G. Dissanayake, and P. W. Gibbens. Toward deployment of large scale simultaneous localisation and map building (SLAM) systems. Technical report, Australian Centre for Field Robotics, University of Sydney, 2000.

[5] J. Guivant and E. Nebot. Optimization of the simultaneous localisation and map building algorithm for real time implementation. *IEEE Trans. Robotics and Automation*, 17, June 2001.

[6] J. Guivant, E. Nebot, and H. Durrant-Whyte. Simultaneous localisation and map building using natural features in outdoor environments. *Intelligent Autonomous Systems*, Feb. 2000.

[7] S. J. Julier and J. K. Uhlmann. Simultaneous localisation and map building using split covariance intersection. In *Proc. IEEE Conf. on Intelligent Robots and Systems, Maui*, Nov. 2001.

[8] J. Knight. Computationally tractable SLAM. Technical Report OUEL 2232/2001, Department of Engineering Science, University of Oxford, 2001.

[9] J. J. Leonard and H. J. S. Feder. Decoupled stochastic mapping. Technical Memorandum 99-1, MIT Marine Robotics Laboratory, Dec. 1999.

[10] P. F. McLauchlan. The variable state dimension filter applied to surface-based structure from motion. CVSSP Technical Report VSSP-TR-4/99, School of Electrical Engineering, Information Technology and Mathematics, University of Surrey, 1999.

[11] P. Newman. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem.* PhD thesis, Australian Centre for Field Robotics, University of Sydney, Mar. 1999.

[12] M. Pollefeys, R. Koch, and L. Van Gool. Self calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. 6th Int'l Conf. on Computer Vision, Bombay*, pages 90–96, 1998.