# Chapter 7
# A Sliding Window Filter for Incremental SLAM

Gabe Sibley, Larry Matthies and Gaurav Sukhatme

## 7.1 Introduction

This work develops a sliding window filter for incremental simultaneous localization and mapping (SLAM) that focuses computational resources on accurately estimating the immediate spatial surroundings using a sliding time window of the most recent sensor measurements. Ideally, we would like a constant time algorithm that closely approximates the all-time maximum-likelihood estimate as well as the minimum variance Cramer Rao lower bound (CRLB) - that is we would like an estimator that achieves some notion of statistical optimality (quickly converges), efficiency (quickly reduces uncertainty) and consistency (avoids over-confidence). To this end we give a derivation of the SLAM problem from the Gaussian non-linear least squares optimization perspective. We find that this results in a simple, yet general, take on the SLAM problem; we think this is a useful contribution.

Our approach is inspired by the results from the photogrammetry community, dating back to the late 1950's [1], and later derivatives like Mikhail's least squares treatment [6], the Variable state dimension filter(VSDF) [5], visual odometry(VO) [4], modern bundle adjustment(BA) [10, 3] and of course extended Kalman filter (EKF) SLAM [9].

We apply the sliding window filter to SLAM with stereo vision and inertial measurements. Experiments show that the best approximate method comes close to matching the performance of the optimal estimator while attaining constant time complexity - empirically, it is often the case that the difference in their performance is indistinguishable.

Gabe Sibley and Gaurav Sukhatme
Robotic and Embedded Systems Laboratory, University of Southern California, Los Angeles California e-mail: gsibley$|$gaurav@usc.edu

Gabe Sibley and Larry Matthies
Jet Propulsion Laboratory, California Institute of Technology in Pasadena California e-mail: gsibley$|$lhm@jpl.nasa.gov

## 7.2 Non-linear Least Squares SLAM

Standard statistical point estimation is a useful tool for understanding the basic structure of the SLAM problem. Non-linear least squares is appealing for a number of reasons. First, because it emphasizes the fundamental minimization principle at work in least squares, which, we would argue, is a principle that is more difficult to see from the recursive estimation perspective. Second, starting with the underlying probability density functions that describe our problem, it clearly shows the basic probabilistic nature of SLAM - that is, SLAM is simply tracking a normal distribution through a large state space; a state space that changes dimension as we undertake the fundamental probabilistic operations of removing parameters via marginalization, and adding parameters via error propagation and conditioning. Another reason to derive SLAM via statistical point estimation is because it exposes a rich body of theory about the convergence of non-linear least squares estimators. With this in mind, we carry forward in the usual way, by describing the system state vector, process model, measurement model and how we incorporate prior information.

### *7.2.1 Parameterization*

The parameter vector is a temporal sequence of robot poses $\mathbf{x}_{\mathbf{p}_j}$, $1 \leq j \leq m$, and 3D landmark positions $\mathbf{x}_{\mathbf{m}_i}$, $1 \leq i \leq n$.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_\mathbf{p} \\ \mathbf{x}_\mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{\mathbf{p}_1} \\ \vdots \\ \mathbf{x}_{\mathbf{p}_m} \\ \mathbf{x}_{\mathbf{m}_1} \\ \vdots \\ \mathbf{x}_{\mathbf{m}_n} \end{bmatrix}$$

A pose $\mathbf{x}_{\mathbf{p}_j}$ at time index $j$ is represented by a six parameter column vector comprised of a 3D point and an Euler angle $\mathbf{x}_{\mathbf{p}_j} = [x_{\mathbf{p}_j}\, y_{\mathbf{p}_j}\, z_{\mathbf{p}_j}\, r_{\mathbf{p}_j}\, p_{\mathbf{p}_j}\, q_{\mathbf{p}_j}]^T$. Map landmarks are represented by their 3D position, $\mathbf{x}_{\mathbf{m}_i} = [x_{\mathbf{m}_i}\, y_{\mathbf{m}_i}\, z_{\mathbf{m}_i}]^T$. The state dimension is thus $|x| = (6m + 3n)$ and grows as the robot path increases and as new landmarks are observed.

### 7.2.2 Kinematic Process Model

The process model $f_j : \mathbb{R}^6 \rightarrow \mathbb{R}^6$ for a single step describes each pose in terms of the previous pose

$$\mathbf{x}_{\mathbf{p}_j} = f_j(\mathbf{x}_{\mathbf{p}_{j-1}}, \mathbf{u}_j) + \mathbf{w}_j \tag{7.1}$$

where $\mathbf{u}_j$ is an input command to the robot. The noise vector $\mathbf{w}_j$ is additive and follows a normal distribution $\mathbf{w}_j \sim \mathcal{N}(0, \mathbf{Q}_j)$. We also assume it is reasonable to have $\mathbf{x}_{\mathbf{p}_j} \sim \mathcal{N}(f_j(\mathbf{x}_{\mathbf{p}_{j-1}}, \mathbf{u}_j), \mathbf{Q}_j)$. A simple and useful kinematic process model for $f_j$ is the compound operation, $\oplus$, which is described in [9]. The 6×6 Jacobian of $f_j$, $\mathbf{F}_j = \left.\frac{\partial f_j}{\partial \mathbf{x}_{\mathbf{p}_j}}\right|_{\mathbf{x}_{\mathbf{p}_j}, \mathbf{u}_{j+1}}$, which we will need in a moment, is also derived in [9]. Concatenating individual process models together, the p.d.f. describing the robot path, $\mathbf{x}_\mathbf{p} = [\mathbf{x}_{\mathbf{p}_1}^T, ..., \mathbf{x}_{\mathbf{p}_m}^T]^T$, is $p(\mathbf{x}_\mathbf{p}) = \mathcal{N}(\mu_\mathbf{p}, \mathbf{Q})$, where

$$\mu_\mathbf{p} = f(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_{\mathbf{p}_1} \\ f_1(\mathbf{x}_{\mathbf{p}_1}, \mathbf{u}_2) \\ \vdots \\ f_m(\mathbf{x}_{\mathbf{p}_{m-1}}, \mathbf{u}_m) \end{bmatrix}, \; \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & & \\ & \ddots & \\ & & \mathbf{Q}_m \end{bmatrix}.$$

In practice, one usually extends this basic model to also estimate other quantities, such as linear and angular velocities; for clarity, we will stick with this basic kinematic formulation.

### 7.2.3 Sensor Model

We say a measurement of the $i^{th}$ landmark taken from the $j^{th}$ pose is related to the state vector by the sensor model $h_{ij} : \mathbb{R}^{|\mathbf{x}_{\mathbf{m}_i}|+|\mathbf{x}_{\mathbf{p}_j}|} \rightarrow \mathbb{R}^{|\mathbf{z}_{ij}|}$

$$\mathbf{z}_{ij} = h_{ij}(\mathbf{x}_{\mathbf{m}_i}, \mathbf{x}_{\mathbf{p}_j}) + \mathbf{v}_{ij} \tag{7.2}$$

which generates the expected value the sensor will return when landmark $i$ is observed from pose $j$. We assume $\mathbf{v}_{ij} \sim \mathcal{N}(0, \mathbf{R}_{ij})$ so that $\mathbf{z}_{ij} \sim \mathcal{N}(h_{ij}, \mathbf{R}_{ij})$, where $\mathbf{R}_{ij}$ is the observation error covariance matrix.

Lumping all the observations, measurement functions and measurement covariances together we write $\mathbf{z}$, $h$, and $\mathbf{R}$ as

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_{11} \\ \vdots \\ \mathbf{z}_{1m} \\ \vdots \\ \mathbf{z}_{nm} \end{bmatrix}, \; h(\mathbf{x}) = \begin{bmatrix} h_{11} \\ \vdots \\ h_{1m} \\ \vdots \\ h_{nm} \end{bmatrix}, \; \mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & 0 & \dots & 0 \\ 0 & \ddots & & \vdots \\ & & \mathbf{R}_{1m} & \\ \vdots & & \ddots & \\ 0 & \dots & & \mathbf{R}_{nm} \end{bmatrix}.$$

Treating the process information as observations, we get the measurement likelihood $p(\mathbf{z}, \mathbf{u}|\mathbf{x}) = \mathcal{N}(\mu_z, \Sigma_z)$, where

$$\mu_z = \begin{bmatrix} h(\mathbf{x}) \\ f(\mathbf{x}) \end{bmatrix}, \; \Sigma_z = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{Q} \end{bmatrix}.$$

### *7.2.4 Point Estimation*

Suppose we are also given *prior information* about the first pose and the map, $p(\mathbf{x}_\pi) = \mathcal{N}\left(\mu_\pi, \Pi^{-1}\right)$, where

$$\mathbf{x}_\pi = \begin{bmatrix} \mathbf{x}_{\mathbf{p}_1} \\ \mathbf{x}_{\mathbf{m}} \end{bmatrix}, \; \mu_\pi = \begin{bmatrix} \hat{\mathbf{x}}_{\mathbf{p}_1} \\ \hat{\mathbf{x}}_{\mathbf{m}} \end{bmatrix}, \; \Pi = \begin{bmatrix} \Pi_{\mathbf{p}_1} & \Pi_{\mathbf{pm}} \\ \Pi_{\mathbf{pm}}^T & \Pi_{\mathbf{m}} \end{bmatrix}.$$

This prior encodes information about a single starting pose, about some previously known map of $n$ landmarks, and about the relationships between the starting pose and the map; $\Pi_{\mathbf{p}}$ is the $6 \times 6$ initial pose information matrix, $\Pi_{\mathbf{m}}$ is $3n \times 3n$ map prior information matrix, and $\Pi_{\mathbf{pm}}$ is the $6 \times 3n$ pose-map information matrix.

Armed with the above we can now write the posterior probability of the system,

$$p(\mathbf{x}|\mathbf{z}, \mathbf{u}) = p(\mathbf{z}, \mathbf{u}|\mathbf{x}) p(\mathbf{x}). \tag{7.3}$$

We wish to compute the *maximum a posteriori* estimate of $\mathbf{x}$ which maximizes this density. First, it helps if we lump the sensor model, process model, and prior information terms together by defining the function $g(\mathbf{x})$ and matrix $\mathbf{C}$ as

$$g(\mathbf{x}) = \begin{bmatrix} g_z(\mathbf{x}) \\ g_f(\mathbf{x}) \\ g_\pi(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{z} - h(\mathbf{x}) \\ \mathbf{x}_{\mathbf{p}} - f(\mathbf{x}_{\mathbf{p}}) \\ \begin{bmatrix} \hat{\mathbf{x}}_{\mathbf{p}_1} \\ \hat{\mathbf{x}}_{\mathbf{m}} \end{bmatrix} - \begin{bmatrix} \mathbf{x}_{\mathbf{p}_1} \\ \mathbf{x}_{\mathbf{m}} \end{bmatrix} \end{bmatrix}, \qquad \mathbf{C}^{-1} = \begin{bmatrix} \mathbf{R}^{-1} & 0 & 0 \\ 0 & \mathbf{Q}^{-1} & 0 \\ 0 & 0 & \Pi \end{bmatrix};$$

then by taking the negative logarithm of (7.3) we get a proportional non-linear least squares problem

$$\ell(\mathbf{x}) = \frac{1}{2} \left( g(\mathbf{x})^T \mathbf{C}^{-1} g(\mathbf{x}) \right). \tag{7.4}$$

Letting $\mathbf{S}^T\mathbf{S} = \mathbf{C}^{-1}$ and $r(\mathbf{x}) = \mathbf{S}g(\mathbf{x})$ then (7.4) is clearly a non-linear least squares problem of the form

$$\ell(\mathbf{x}) = \frac{1}{2}||r(\mathbf{x})||^2. \tag{7.5}$$

Newton's solution to such optimization problems is the iterative sequence

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (\nabla^2\ell(\mathbf{x}_i))^{-1}\nabla\ell(\mathbf{x}_i). \tag{7.6}$$

For small residual problems a useful approximation to (7.6) is the Gauss-Newton method, which approximates the Hessian $\nabla^2\ell(\mathbf{x}_i)$ by $r'(\mathbf{x}_i)^T r'(\mathbf{x}_i)$. Thus, since the gradient of (7.5) is $\nabla\ell(\mathbf{x}_i) = r'(\mathbf{x}_i)^T r(\mathbf{x}_i)$, the Gauss-Newton method defines the sequence of iterates [2]

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (r'(\mathbf{x}_i)^T r'(\mathbf{x}_i))^{-1} r'(\mathbf{x}_i)^T r(\mathbf{x}_i) \tag{7.7}$$

Noting that $r'(\mathbf{x}_i) = \mathbf{S}\mathbf{G}_i$ where $\mathbf{G}_i$ is the Jacobian of $g(\mathbf{x}_i)$, (7.7) becomes

$$\delta\mathbf{x}_i = (\mathbf{G}_i^T\mathbf{C}^{-1}\mathbf{G}_i)^{-1}\mathbf{G}_i^T\mathbf{C}^{-1}g(\mathbf{x}_i). \tag{7.8}$$

such that $\mathbf{x}_{i+1} = \mathbf{x}_i + \delta\mathbf{x}_i$. When iterated, this sequence is locally q-quadratically convergent to the MAP estimate for near zero-residual problems [2]. The system of linear equations

$$\mathbf{G}_i^T\mathbf{C}^{-1}\mathbf{G}_i\delta\mathbf{x}_i = \mathbf{G}_i^T\mathbf{C}^{-1}g(\mathbf{x}_i) \tag{7.9}$$

is the essential least squares form of the SLAM problem (we will often omit the iteration index). The difference between many SLAM algorithms can be boiled down to differences in how these equations are solved. It is also interesting to note here that for many problems the Gauss-Newton method is algebraically identical to the iterated extended Kalman filter (IEKF).

## 7.3 Sparsity in the System Equations

Before describing the sliding window filter it is useful to take a look at the overall structure of the SLAM least squares equations, and to study how this structure lends itself to various algebraic solutions.

Expanding the Jacobian $\mathbf{G}$,

$$\mathbf{G} = \begin{bmatrix} \frac{\partial g_z}{\partial \mathbf{x}} \\ \frac{\partial g_f}{\partial \mathbf{x}} \\ \frac{\partial g_\pi}{\partial \mathbf{x}} \end{bmatrix} = - \begin{bmatrix} \mathbf{H} \\ \mathbf{D} \\ \mathbf{L} \end{bmatrix},$$

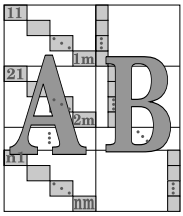


**Fig. 7.1** Basic structure of the sensor model Jacobian, $\mathbf{H}$.

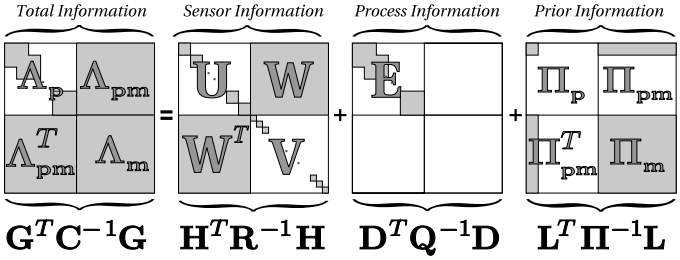


**Fig. 7.2** The sparse structure of least squares SLAM system matrix is due to contributions from three components: the measurement block $\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}$, the process block $\mathbf{D}^T\mathbf{Q}^{-1}\mathbf{D}$, and the prior information block $\mathbf{L}^T\Pi\mathbf{L}$.

we see that the system matrix, $\mathbf{G}^T\mathbf{C}^{-1}\mathbf{G} = \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} + \mathbf{D}^T\mathbf{Q}^{-1}\mathbf{D} + \mathbf{L}^T\Pi\mathbf{L}$, has a sparse structure. The structure of $\mathbf{H}$ is shown in Fig. 7.1. The sparsity pattern of least squares SLAM system matrix is due to contributions from the three components

$$\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} = \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix}, \; \mathbf{D}^T\mathbf{Q}^{-1}\mathbf{D} = \begin{bmatrix} \mathbf{E} & 0 \\ 0 & 0 \end{bmatrix}, \text{ and } \mathbf{L}^T\Pi\mathbf{L} = \begin{bmatrix} \Pi_{\mathbf{p}} & 0 & \dots & \Pi_{\mathbf{pm}} \\ 0 & 0 & & 0 \\ \vdots & 0 & \ddots & \vdots \\ \Pi_{\mathbf{pm}}{}^T & 0 & \dots & \Pi_{\mathbf{m}} \end{bmatrix}$$

where $\mathbf{U} = \mathbf{A}^T\mathbf{R}^{-1}\mathbf{A}$, $\mathbf{W} = \mathbf{A}^T\mathbf{R}^{-1}\mathbf{B}$ and $\mathbf{V} = \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}$. The block tri-diagonal process matrix is

$$\mathbf{E} = \begin{bmatrix} \mathbf{Q}_1^{-1} + \mathbf{F}_1\mathbf{Q}_2^{-1}\mathbf{F}_1^T & -\mathbf{F}_1^T\mathbf{Q}_2^{-1} & 0 & \dots & 0 \\ -\mathbf{Q}_2^{-1}\mathbf{F}_1 & \mathbf{Q}_2^{-1} + \mathbf{F}_2\mathbf{Q}_3^{-1}\mathbf{F}_2^T & \ddots & & \vdots \\ 0 & \ddots & \ddots & & 0 \\ \vdots & & & \mathbf{Q}_{m-1}^{-1} + \mathbf{F}_{m-1}\mathbf{Q}_m^{-1}\mathbf{F}_{m-1}^T & -\mathbf{F}_{m-1}^T\mathbf{Q}_m^{-1} \\ 0 & \dots & 0 & -\mathbf{Q}_m^{-1}\mathbf{F}_{m-1} & \mathbf{Q}_m^{-1} \end{bmatrix}.$$

This structure is also depicted graphically in Fig. 7.2. The task is to solve the system of normal equations 7.7 which expand to

$$\begin{bmatrix} \Lambda_p & \Lambda_{pm} \\ \Lambda_{pm}{}^T & \Lambda_m \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{\mathbf{p}} \\ \delta\mathbf{x}_{\mathbf{m}} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{\mathbf{p}} \\ \mathbf{g}_{\mathbf{m}} \end{bmatrix}$$

where $\mathbf{g}_{\mathbf{p}}$ and $\mathbf{g}_{\mathbf{m}}$ are the least squares RHS vector corresponding to the robot path and map, respectively. We solve this system of equations using elementary matrix operations - for example the Schur complement - to *reduce* the lower right map block $\Lambda_m$ onto the upper left process block $\Lambda_p$

$$\begin{bmatrix} \Lambda_p - \Lambda_{pm}(\Lambda_m)^{-1}\Lambda_{pm}{}^T & 0 \\ \Lambda_{pm}{}^T & \Lambda_m \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{\mathbf{p}} \\ \delta\mathbf{x}_{\mathbf{m}} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{\mathbf{p}} - \Lambda_{pm}(\Lambda_m)^{-1}\mathbf{g}_{\mathbf{m}} \\ \mathbf{g}_{\mathbf{m}} \end{bmatrix}$$

which is solved directly for $\delta\mathbf{x_p}$ and then for $\delta\mathbf{x_m}$ by back-substitution:

$$\begin{aligned}\delta\mathbf{x_p} &= (\Lambda_p - \Lambda_{pm}(\Lambda_m)^{-1}{\Lambda_{pm}}^T)^{-1}(g_p - \Lambda_{pm}(\Lambda_m)^{-1}g_\mathbf{m}) \\ \delta\mathbf{x_m} &= (\Lambda_m)^{-1}(g_\mathbf{m} - {\Lambda_{pm}}^T\delta\mathbf{x_p})\end{aligned}$$

Alternately, we can also reduce the upper left process block $\Lambda_p$ onto the lower right map block $\Lambda_m$

$$\begin{bmatrix}\Lambda_p & \Lambda_{pm} \\ 0 & \Lambda_m - {\Lambda_{pm}}^T(\Lambda_p)^{-1}\Lambda_{pm}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x_p} \\ \delta\mathbf{x_m}\end{bmatrix} = \begin{bmatrix}\mathbf{g_p} \\ \mathbf{g_m} - {\Lambda_{pm}}^T(\Lambda_p)^{-1}\mathbf{g_p}\end{bmatrix}$$

giving the solution

$$\begin{aligned}\delta\mathbf{x_m} &= (\Lambda_m - {\Lambda_{pm}}^T(\Lambda_p)^{-1}\Lambda_{pm})^{-1}(g_\mathbf{m} - {\Lambda_{pm}}^T(\Lambda_p)^{-1}g_p) \\ \delta\mathbf{x_p} &= (\Lambda_{pm})^{-1}(g_p - \Lambda_{pm}\delta\mathbf{x_m})\end{aligned}$$

Depending on the process noise and the prior, the system matrix $\mathbf{G}^T\mathbf{C}^{-1}\mathbf{G}$ can take on different sparsity patterns that affect the complexity of finding a solution. In the field, the problem at hand will define the sparsity pattern, which will influence the choice of which algorithm to use.

## 7.4 The Sliding Window Filter

To keep the complexity of the filter constant with the number of landmarks it is necessary to reduce the size of the state vector. This is accomplished by removing the oldest pose parameters and distant landmark parameters. If we directly remove parameters from the system equation however, we can lose information about how the parameters interact. The right way to remove parameters from a multi-dimensional normal distribution is to marginalize them out.

### *7.4.1 The Effects of Marginalizing Out Parameters*

Marginalizing out a set of pose parameters will add cross-information terms in the SLAM least squares system matrix (that is the Hessian, or information matrix) between all the landmarks that were conditionally dependent on those parameters. This is depicted graphically in Fig. 7.3 for a system that starts *without* any prior information. Studying this structure we see that downdating the oldest pose causes fill-in in three places: 1) between any landmarks that were visible from the downdated pose, 2) between the parameters of the next-oldest-pose (the pose one time step after the pose being downdated), and 3) between the next-oldest-pose and all landmarks seen by the downdated pose. Interestingly, $\Pi$ is the only place that ever suffers from fill-in. Because of this structure, when solving we can still take advantage of any

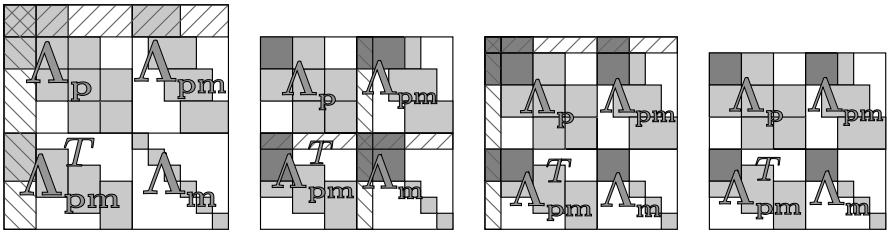


**Fig. 7.3** Information matrix evolution for an example problem with 4 poses and 6 landmarks. The left image is after measuring landmarks 1, 2, 3 from pose 1, landmarks 2, 3, and 4 from pose 2, landmarks 3, 4, and, 5 from pose 3 and 4, 5, and 6 at pose 4. In the second image from left we see that marginalizing out pose 1 induces conditional dependencies (fill-in) in three places: 1) the top left $6 \times 6$ of the process-block, 2) the prior map-block $\Pi_m$ between landmarks that were visible from pose 1, and 3) the prior pose-to-map block $\Pi_{pm}$ between landmarks that were visible from pose 1. These places are shaded in darker grey. At this point (second from right image) downdating landmark 1, which is not visible from any of the remaining poses, will induce no extra fill-in in $\Pi$ (right image).

sparsity patterns in $\Lambda_{pm}$ and $\Lambda_p$. It is important to note that the $\Pi$ term catches all the prior information as we "roll" up old state parameters. If we were to ignore $\Pi$, we would not benefit from past measurements. Marginalizing out landmarks that are not visible from any active pose will also only ever cause fill in $\Pi$.

Marginalizing out poses at a fixed rate and landmarks when they lose support results in a constant time complexity incremental SLAM estimation algorithm. By choosing when to downdate poses and landmarks sliding window SLAM can scale from the full batch solution, to the extended Kalman filter solution. That these algorithms are subsumed within one framework testifies to the generality of the simple least squares approach.

It is interesting to note what happens if we simply delete parameters from the estimator instead of marginalizing them out. For a sliding window of size $k$, the error converges like $1/k$ just as we would expect the batch estimator to do. However, after $k$ steps, the error stops converging as we delete information from the back of the filter. With such deleting and a sliding window of $k = 2$ it is interesting to note that we end up with a solution that is nearly identical to previous forms of Visual Odometry [4, 7, 8]. The graph in Fig. 7.4 shows the average RMS mapping error for this type of Visual Odometry compared to the batch solution, as well as the sliding window filter solution.

## 7.5 Conclusions

This chapter describes a SLAM solution that concentrates computational resources on accurately estimating the immediate spatial surroundings by using a sliding time window of the most recent sensor measurements. Focusing computation on improving the local result is crucial for applications that wish to fuse spatially high-

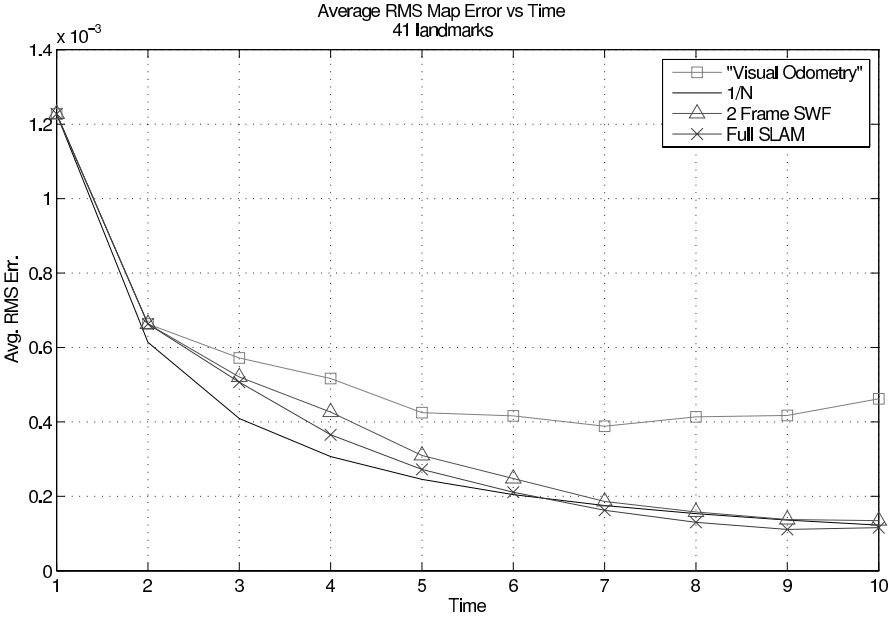


**Fig. 7.4** Graph showing average RMS mapping error. Each curve is a trial for different size time window, averaged over 50 Monte-Carlo trials, with 0.1 pixel std. dev. measurement noise. 1.0m std. dev. process noise. Note that the sliding window filter comes close to the full SLAM solution. A sliding window of 2 is close to optimal $1/k$ full batch curve. Further, because VO does not combine information over time, it does not reduce uncertainty as time passes.

resolution, dense structure estimates. With high bandwidth sensors (like cameras) this is clearly beneficial for computational reasons, and it especially true if we wish to fuse all of the sensor data (or a significant portion thereof).

By tuning a few parameters, the sliding window algorithm can scale from exhaustive batch solutions to fast incremental solutions. Ideally, we would like a constant time algorithm that closely approximates the all-time maximum-likelihood estimate as well as the minimum variance Cramer Rao Lower Bound - that is, we would like an estimator that achieves some notion of statistical optimality (quickly converges), efficiency (quickly reduces uncertainty) and consistency (avoids over-confidence). We find that approaching this problem from the statistical point estimation point of view results in a simple, yet general, take on the SLAM problem; we think this is a useful contribution. Data-fusion is fundamental for improving a robot's metric estimation of the world. Doing it quickly and with large amounts of data is a challenging task. Ultimately, some form of dense data-fusion will enable accurate high-resolution spatial perception for autonomous robots.

**Acknowledgements** This work is supported in part by Caltech/JPL under contract 1277958, and by NSF grants IIS-0133947 and CCR-0120778.

## References

1. Brown, D.: A solution to the general problem of multiple station analytical stereotriangulation. Tech. rep., RCP-MTP Data Reduction Technical Report No. 43, Patrick Air Force Base, Flordia (also designated as AFMTC 58-8) (1958)
2. Dennis, J.J., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for Industrial & Applied Mathematics (1996)
3. Engels, C., Stewenius, H., Nister, D.: Bundle adjustment rules. In: Photogrammetric Computer Vision (2006)
4. Matthies, L., Shafer, S.: Error modelling in stereo navigation. IEEE Journal of Robotics and Automation **3**(3), 239–248 (1987)
5. McLauchlan, P.F.: The variable state dimension filter applied to surface-based structure from motion. Tech. rep., University of Surrey (1999)
6. Mikhail, E.M.: Observations and Least Squares. Rowman & Littlefield (1983)
7. Nister, D., Naroditsky, O., Bergen, J.: Visual odometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–659. Washington, DC (2004)
8. Olson, C.F., Matthies, L.H., Schoppers, M., Maimone, M.W.: Stereo ego-motion improvements for robust rover navigation. In: Proceedings of the IEEE Conference on Robotics and Automation, pp. 1099–1104. Washington, DC (2001)
9. Smith, R.C., Self, M., Cheeseman, P.: Estimating uncertain spatial relationships in robotics. In: I.J. Cox, G.T. Wilfong (eds.) Autonomous Robot Vehicles, pp. 167–193. Springer-Verlag (1990)
10. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment – A modern synthesis. In: W. Triggs, A. Zisserman, R. Szeliski (eds.) Vision Algorithms: Theory and Practice, LNCS, pp. 298–375. Springer Verlag (2000)