Amortized Constant Time State Estimation in Pose SLAM and Hierarchical SLAM using a Mixed Kalman-Information Filter

Viorela Ila*, Josep M. Porta, Juan Andrade-Cetto

Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain

Abstract

The computational bottleneck in all information-based algorithms for simultaneous localization and mapping (SLAM) is the recovery of the state mean and covariance. The mean is needed to evaluate model Jacobians and the covariance is needed to generate data association hypotheses. In general, recovering the state mean and covariance requires the inversion of a matrix with the size of the state, which is computationally too expensive in time and memory for large problems. Exactly sparse state representations, such as that of Pose SLAM, alleviate the cost of state recovery either in time or in memory, but not in both. In this paper, we present an approach to state estimation that is linear both in execution time and in memory footprint at loop closure, and constant otherwise. The method relies on a state representation that combines the Kalman and the information-based approaches. The strategy is valid for any SLAM system that maintains constraints between marginal states at different time slices. This includes both Pose SLAM, the variant of SLAM where only the robot trajectory is estimated, and hierarchical techniques in which submaps are registered with a network of relative geometric constraints.

Keywords: State recovery, Kalman filter, Information filter, Pose SLAM, Hierarchical SLAM

1. Introduction

Seminal solutions to the simultaneous localization and mapping (SLAM) problem relied on the extended Kalman filter (EKF) to estimate the mean absolute position of landmarks and the robot pose and their associated covariance matrix [1]. This has quadratic memory and computational cost, limiting its use to small areas.

Instead of using the mean and the covariance, Gaussian distributions can be parametrized in canonical form using the information vector and the information matrix. In SLAM, the information matrix turns out to be approximately sparse, i.e., the matrix entries for distant landmarks are very small and the matrix can be sparsified with a minimal information loss, trading optimality for efficiency [2]. Efficiency without information loss is possible by estimating the entire robot path along with the map, an approach typically referred to as full SLAM [3–5]. Exact sparsification is also possible if only a set of variables is maintained; either by keeping a small set of active landmarks [6], by decoupling the estimation problem maintaining the map only [7], or as it is done in Pose SLAM, by maintaining only the pose history [8, 9]. In Pose SLAM, landmarks are only used to obtain relative measurements linking pairs of poses. When working with sensors that are able to identify many landmarks per pose, Pose SLAM produces more compact maps than the other exactly sparse approaches.

Due to their small memory footprint, sparse representations enable SLAM solutions that scale nicely to very large maps. Off-line information-based SLAM approaches [4, 10, 11] obtain the maximum likelihood solution from the constraints encoded in the information matrix. The optimization iteratively approximates the mean by solving a sequence of linear systems using the previously estimated mean as a linearization point for the constraints. This process assumes data association for granted, somehow limiting its applicability. On-line information-based approaches rely either on variants of the batch methods [5] or, more commonly, on filtering [8, 12] using the Extended Information Filter (EIF) as the estimation tool of choice. These on-line

*Corresponding author.
Email addresses: vila@iri.upc.edu (Viorela Ila*), porta@iri.upc.edu (Josep M. Porta), cetto@iri.upc.edu (Juan Andrade-Cetto)

systems not only have to recover the mean to evaluate the Jacobians, but also need to address the data association problem. Data association might be tackled directly from sensor readings, without relying on the filtered pose priors [13]. The process, however, is prone to perceptual aliasing and it is often convenient to take advantage of the state estimates to limit the search space. False positives can be avoided performing prior-based data association tests that use cross covariances between match candidates. Neither, the mean or the cross covariances, are directly available from the estimates of the information-based representations.

The EKF and the EIF applied to SLAM are different in nature. While in the former the estimate includes all the necessary data for linearization and data association, the latter is advantageous from the point of view of memory footprint. In this paper, we propose a combination of these two filters with the aim of getting the best of the two worlds: reduced memory complexity and easy access to the mean and the relevant blocks of the covariance matrix.

The work presented in this paper improves the formalization of the state estimation technique in [14], where we adopted an extended information filter approach. Here, we abandon this paradigm and propose a novel mixed Kalman-information filter. Moreover, while the approach presented in [14] is limited to Pose SLAM, here we exploit the properties of the new mixed Kalman-information filter to generalize the approach to both the Pose SLAM problem and to hierarchical SLAM. For the sake of clarity, our presentation is sequential in order, first we introduce the new approach in the context of Pose SLAM and latter we extend it to hierarchical SLAM.

The paper is structured as follows. In Section 2, we formalize the Pose SLAM problem and describe its solution via EKF and EIF. In Section 3, we describe a combination of the two filters that allows state estimation in linear time and space complexities. Section 4 describes a refinement of the presented approach that allows updates in constant time during open loop traverse. This is relevant for approaches that carefully select the loops to close in order to avoid inconsistency as much as possible [12] or where previously mapped areas are barely revisited [15]. In these contexts, the linear time complexity of loop closure is amortized over long periods yielding an almost constant time state update. Section 5 extends the approach to hierarchical SLAM and Section 6 presents results both with simulated data and with real data sets that validate the presented approach. Concluding remarks are given in Section 7.

2. Pose SLAM Formulation

In the on-line form of Pose SLAM, the objective is to estimate the trajectory of the robot, $\mathbf{x}_n = \{x_0, \ldots, x_n\}$, with x_i the robot pose at time i. The following applies for poses in $SE(2)$ or in $SE(3)$ and in Section 6 we particularize the approach to the planar case. Using a Bayesian recursion, the trajectory, $\mathbf{x}_n$, is updated given a set of observations, $\mathbf{z}_n$, of the relative displacement between the current robot pose and previous poses along the path

$$p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n) \propto p(\mathbf{x}_n|\mathbf{x}_{n-1})\, p(\mathbf{z}_n|\mathbf{x}_n).$$

The observations set $\mathbf{z}_n$ can be split in two independent groups $\mathbf{z}_n = \{\mathbf{u}_n, \mathbf{y}_n\}$ where $\mathbf{u}_n$ gives the displacement between the current robot pose and the immediate previous one, and $\mathbf{y}_n$ links the current pose with any other pose but the previous one. With this, the probabilistic model becomes

$$\begin{aligned} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n) &\propto p(\mathbf{x}_n|\mathbf{x}_{n-1})\, p(\mathbf{u}_n, \mathbf{y}_n|\mathbf{x}_n) \\ &\propto p(\mathbf{x}_n|\mathbf{x}_{n-1})\, p(\mathbf{u}_n|\mathbf{x}_n)\, p(\mathbf{y}_n|\mathbf{x}_n) \\ &\propto p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{u}_n)\, p(\mathbf{y}_n|\mathbf{x}_n). \end{aligned} \quad (1)$$

The estimation problem in Eq. (1) corresponds to the SLAM operations of augmenting the state, $p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{u}_n)$, and updating the robot path using relative observations, $p(\mathbf{y}_n|\mathbf{x}_n)$.

Assuming Gaussian distributions, the probabilities in Eq. (1) can be parametrized either in terms of their mean and covariance, $\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, or in terms of the information vector and matrix, $\mathbf{x}_n \sim \mathcal{N}^{-1}(\boldsymbol{\eta}_n, \boldsymbol{\Lambda}_n)$, with $\boldsymbol{\eta}_n = \boldsymbol{\Lambda}_n \boldsymbol{\mu}_n$, $\boldsymbol{\Lambda}_n = \boldsymbol{\Sigma}_n^{-1}$, and in which the estimation workhorses are the extended Kalman and information filters, respectively.

Note that simultaneous observations are independent and, thus, observations linking the same pair of poses can be fused before using them to update the filter. In particular, we can assume the set $\mathbf{u}_n$ to include a single element, u_n.

2.1. EKF Pose SLAM State Estimation

The observation $u_n \sim \mathcal{N}(\mu_u, \boldsymbol{\Sigma}_u)$ is used to augment the state with a new pose. In Pose SLAM, the state transition model is given by

$$\begin{aligned} x_n &= f(x_{n-1}, u_n) \\ &\approx f(\mu_{n-1}, \mu_u) + \mathbf{F}_n\, (x_{n-1} - \mu_{n-1}) + \mathbf{W}_n(u_n - \mu_u) \end{aligned}$$

with $\mathbf{F}_n$ and $\mathbf{W}_n$ the Jacobians of f with respect to x_{n-1} and u_n, evaluated at μ_{n-1} and μ_u. The EKF augments the

state as

$$\boldsymbol{\mu}_n = \begin{bmatrix} \boldsymbol{\mu}_{n-1} \\ x_n \end{bmatrix}, \tag{2}$$

$$\boldsymbol{\Sigma}_n = \begin{bmatrix} \boldsymbol{\Sigma}_{1:n-1,1:n-1} & \boldsymbol{\Sigma}_{1:n-1,n-1}\,\mathbf{F}_n^\top \\ \mathbf{F}_n\,\boldsymbol{\Sigma}_{n-1,1:n-1} & \mathbf{F}_n\,\boldsymbol{\Sigma}_{n-1\,n-1}\,\mathbf{F}_n^\top + \mathbf{Q} \end{bmatrix}, \tag{3}$$

with $\mathbf{Q} = \mathbf{W}_n \boldsymbol{\Sigma}_u \mathbf{W}_n^\top$ and where $\boldsymbol{\Sigma}_{n-1\,n-1}$ is used to denote the block of $\boldsymbol{\Sigma}_{n-1}$ corresponding to the $(n-1)$-th pose, and $\boldsymbol{\Sigma}_{1:k,1:k}$ indicate the blocks ranging from the first to the k-th pose.

Each set of measures $\mathbf{y}_n = \{y_n^i, \ldots, y_n^k\}$ constrains the relative position of the last pose to some other poses from the robot trajectory forming loops. The measurement model for each of these constraints is

$$\begin{aligned} y_n^i &= h(x_i, x_n) + \mathbf{v}_n \\ &\approx h(\mu_i, \mu_n) + \mathbf{H}(\mathbf{x}_n - \boldsymbol{\mu}_n) + \mathbf{v}_n, \end{aligned}$$

where h gives the displacement from x_i to x_n in the reference frame of x_i, and $\mathbf{H}$ is

$$\mathbf{H} = [\mathbf{0}\ \ldots \mathbf{0}\ \mathbf{H}_i\ \mathbf{0}\ \ldots\ \mathbf{0}\ \mathbf{H}_n], \tag{4}$$

with $\mathbf{H}_i$ and $\mathbf{H}_n$ the Jacobians of h with respect to x_i and x_n, and $\mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_y)$ the measurement white noise.

The information from observation y_n^i is merged into the filter applying the following increments

$$\Delta\boldsymbol{\mu} = \mathbf{K}\,(y_n^i - h(\mu_i, \mu_n)), \tag{5}$$

$$\Delta\boldsymbol{\Sigma} = -\mathbf{K}\,\mathbf{H}\,\boldsymbol{\Sigma}_n, \tag{6}$$

to $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$, respectively, where $\mathbf{K}$ is the Kalman gain, $\mathbf{K} = \boldsymbol{\Sigma}_n\,\mathbf{H}^\top\,\mathbf{S}^{-1}$, and $\mathbf{S}$ the innovation matrix, $\mathbf{S} = \mathbf{H}\,\boldsymbol{\Sigma}_n\,\mathbf{H}^\top + \boldsymbol{\Sigma}_y$.

Measurements y_n^i result from the data association process. Instead of directly comparing the sensor readings for the current pose with those for all poses along the trajectory, data association is generally tested on a limited region of the trajectory. To identify poses that are close enough to the current one so that the corresponding sensor readings are likely to match (i.e., to produce y_n^i observations), we can estimate the relative displacement, d, from the current robot pose, x_n, to any other previous pose in the trajectory, x_i, as a Gaussian with parameters

$$\mu_d = h(\mu_i, \mu_n), \tag{7}$$

$$\boldsymbol{\Sigma}_d = [\mathbf{H}_i\ \mathbf{H}_n] \begin{bmatrix} \boldsymbol{\Sigma}_{ii} & \boldsymbol{\Sigma}_{in} \\ \boldsymbol{\Sigma}_{in}^\top & \boldsymbol{\Sigma}_{nn} \end{bmatrix} [\mathbf{H}_i\ \mathbf{H}_n]^\top, \tag{8}$$

where $\boldsymbol{\Sigma}_{in}$ is the cross correlation between the i-th and the current poses. Only poses whose relative displacement, d, is likely to be inside sensor range need to be considered for sensor registration.

Whereas the EKF estimation maintains all the data necessary for linearization and for data association, its drawback is that storing and updating the whole covariance matrix entails quadratic cost both in memory and in execution time.

2.2. EIF Pose SLAM State Estimation

In the EIF form of Pose SLAM [8], the state is augmented as

$$\boldsymbol{\eta}_n = \begin{bmatrix} \boldsymbol{\eta}_{1:n-2} \\ \eta_{n-1} - \mathbf{F}_n{}^\top \mathbf{Q}^{-1}\,(f(\mu_{n-1}, \mu_u) - \mathbf{F}_n\,\mu_{n-1}) \\ \mathbf{Q}^{-1}\,(f(\mu_{n-1}, \mu_u) - \mathbf{F}_n\,\mu_{n-1}) \end{bmatrix},$$

$$\boldsymbol{\Lambda}_n = \begin{bmatrix} \boldsymbol{\Lambda}_{1:n-2,1:n-2} & \boldsymbol{\Lambda}_{1:n-2,n-1} & \mathbf{0} \\ \boldsymbol{\Lambda}_{n-1,1:n-2} & \boldsymbol{\Lambda}_{n-1\,n-1} + \mathbf{F}_n^\top \mathbf{Q}^{-1} \mathbf{F}_n & -\mathbf{F}_n^\top \mathbf{Q}^{-1} \\ \mathbf{0} & -\mathbf{Q}^{-1}\mathbf{F}_n & \mathbf{Q}^{-1} \end{bmatrix}. \tag{9}$$

The information from observation y_n^i is fed to the filter by adding the following increments

$$\begin{aligned} \Delta\boldsymbol{\eta} &= \mathbf{H}^\top\,\boldsymbol{\Sigma}_y^{-1}\,(y_n^i - h(\mu_i, \mu_n) + \mathbf{H}\,\boldsymbol{\mu}_n), \\ \Delta\boldsymbol{\Lambda} &= \mathbf{H}^\top \boldsymbol{\Sigma}_y^{-1} \mathbf{H}, \end{aligned} \tag{10}$$

to $\boldsymbol{\eta}_n$ and $\boldsymbol{\Lambda}_n$, respectively.

Equation (9) defines a block-tridiagonal matrix and Eq. (10) only adds off-diagonal elements to the positions corresponding to the two poses directly related by the observation y_n^i. In practice, due to limited sensor range, only few nearby poses are related and $\boldsymbol{\Lambda}$ remains sparse and, therefore, the memory requirements for the information-based representation can be considered linear with the number of poses.

Notice, however, that the Jacobians above have to be evaluated at the state mean which is not directly available in information form. Moreover, the displacement measure in Eqs. (7) and (8) requires marginalizing out some blocks of the covariance matrix (its block diagonal and the last column) which are also not available in the information form.

The mean can be recovered by solving the following linear system

$$\boldsymbol{\Lambda}_n\,\boldsymbol{\mu}_n = \boldsymbol{\eta}_n\,.$$

Using sparse Cholesky factorization [16], this can be solved in linear time, with the number of non-zeros entries in $\boldsymbol{\Lambda}$. Thus, it scales linearly with the number of poses, except for densely connected degenerate cases.

The covariance can be recovered solving

$$\boldsymbol{\Lambda}_n\,\boldsymbol{\Sigma}_n = \mathbf{I},$$

with $\mathbf{I}$ the identity matrix. Sparse Cholesky factorization allows to solve this system efficiently but with

quadratic memory cost for storing full covariance matrix $\mathbf{\Sigma}_n$. This cost can be alleviated by solving n independent systems, one for each block column of the covariance matrix, $\mathbf{T}_i$, $i \in \{1, \ldots, n\}$

$$\mathbf{\Lambda}_n \, \mathbf{T}_i = \mathbf{I}_i, \tag{11}$$

where $\mathbf{I}_i$ is the sparse block column matrix with an identity block at the position corresponding to pose i. In this way, space complexity is linear, but time complexity is increased due to the overhead of solving many linear systems.

3. Pose SLAM with a Mixed Kalman-Information Representation

To obtain a state recovery strategy that scales linearly both in execution time and in memory usage we propose a mixed Kalman-information representation. We store the state mean, $\boldsymbol{\mu}_n$, a block-vector $\mathbf{D}_n$ containing the block diagonal terms of the covariance matrix, the block-last column of the covariance matrix, $\mathbf{T}_n$, and the information matrix $\mathbf{\Lambda}_n$. The mean is used to evaluate Jacobians, $\mathbf{D}_n$ and $\mathbf{T}_n$ are used for data association, and $\mathbf{\Lambda}_n$ stores in a very compact way the full set of correlations between all poses, which are necessary to propagate the effects of each loop closure to the entire trajectory. Neither the rest of entries in $\mathbf{\Sigma}_n$ nor the information vector $\boldsymbol{\eta}_n$ are maintained. The largest stored element is $\mathbf{\Lambda}_n$, which in practice scales linearly with the number of poses. Therefore, the whole representation scales linearly.

During state augmentation, $\boldsymbol{\mu}_n$ grows as shown in Eq. (2). The new blocks of $\mathbf{D}_n$ and $\mathbf{T}_n$ are computed using the corresponding parts from Eq. (3) and $\mathbf{\Lambda}_n$ is extended as in Eq. (9). Augmenting the mean block-vector, $\boldsymbol{\mu}_n$, and the block-vector of diagonal covariance entries, $\mathbf{D}_n$, can be done in constant time. However, state augmentation produces a new block column $\mathbf{T}_n$ of cross covariance terms with $n-1$ elements. Therefore updating $\mathbf{T}_n$ has linear computational cost. Linear complexity data association can be carried out at the same time $\mathbf{T}_n$ is updated.

When establishing a link between the last pose and any i-th pose from the trajectory, due to the sparse form of the Jacobian $\mathbf{H}$ in Eq. (4), Eqs. (5) and (6) do not require all entries of the covariance matrix but only $\mathbf{D}_n$, $\mathbf{T}_n$ and the i-th block columns of the covariance matrix, $\mathbf{T}_i$. However, $\mathbf{T}_i$ can be obtained in linear time in the number of non-null entries in $\mathbf{\Lambda}_n$ by solving the system in Eq. (11).

The last step is to update the state mean, the block diagonal terms of the covariance, and the last block column of the covariance matrix. Applying Cholesky decomposition to the inverse of the Kalman innovation $\mathbf{S}^{-1} = \mathbf{V}^\top \mathbf{V}$ we define the block column matrix

$$\mathbf{B} = \mathbf{\Sigma}_n \, \mathbf{H}^\top \mathbf{V}^\top,$$

that considering Eq. (4) becomes

$$\mathbf{B} = [\mathbf{T}_i \; \mathbf{T}_n] \begin{bmatrix} \mathbf{H}_i^\top \\ \mathbf{H}_n^\top \end{bmatrix} \mathbf{V}^\top.$$

With this, the mean is updated as in Eq. (5) with $\mathbf{K} = \mathbf{B} \, \mathbf{V}$. The block diagonal entries of the new covariance matrix are updated adding the following increment to $\mathbf{D}_n$

$$\Delta \mathbf{D} = -\begin{bmatrix} \mathbf{B}_1 \, \mathbf{B}_1^\top \\ \vdots \\ \mathbf{B}_n \, \mathbf{B}_n^\top \end{bmatrix},$$

where $\mathbf{B}_i$ is the i-th block row of $\mathbf{B}$. Moreover, $\mathbf{T}_n$ is updated with

$$\Delta \mathbf{T} = -\mathbf{B} \, \mathbf{B}_n^\top.$$

Finally, the information matrix is updated as in Eq. (10).

This process is applied for each loop closed at a same time slice. In practice, and due to sensor limitations, a bounded number of loops per step are closed and, therefore, the whole state update process scales linearly in time and memory with the number of steps.

The new approach avoids quadratic memory requirements of an EKF by maintaining the correlations in information form while, at the same time, allowing direct access to the mean and the covariance entries. In this way, the proposed filtering scheme gets the best of the two worlds; direct marginalization available in covariance form and efficiency from sparse information representation.

4. Open Loop State Recovery in Constant Time

The mixed Kalman-information approach presented above can be applied regardless of the number of asserted loop closures, giving linear time execution per time slice. However, in many cases loops are scarcely closed. For instance in Pose SLAM, in order to delay filter inconsistency as much as possible, it is desired to close only highly informative loops [12]. In this case, the robot operates most of the time in exploration mode, where the most expensive step is that of updating the last block column of the covariance matrix, $\mathbf{T}_n$. This last block column is composed by the cross covariances

$\boldsymbol{\Sigma}_{in}$ between the current robot pose, x_n, and any previous pose x_i, with $i < n$. As described in Eq. (3), when the robot operates in open loop these cross covariances are updated as

$$\boldsymbol{\Sigma}_{in} = \boldsymbol{\Sigma}_{in-1}\, \mathbf{F}_n^\top .$$

Let us assume that a loop is closed at time l. At that point, $\mathbf{T}_l$ includes all marginal $\boldsymbol{\Sigma}_{il}$ for $i \leq l$ and is obtained using Eq. (11). With that, we can unfold the recursive relation and factorize $\boldsymbol{\Sigma}_{in}$ as

$$\boldsymbol{\Sigma}_{in} = \boldsymbol{\Phi}_i\, \mathbf{L}_i\, \mathbf{F}\, \mathbf{F}_n^\top$$

with

$$\boldsymbol{\Phi}_i = \begin{cases} \boldsymbol{\Sigma}_{il} & 1 \leq i \leq l, \\ \boldsymbol{\Sigma}_{ii} & l < i < n, \end{cases} \qquad (12)$$

$$\mathbf{L}_i = \begin{cases} \mathbf{I} & 1 \leq i \leq l, \\ (\mathbf{F}_{l+1}^\top \dots \mathbf{F}_i^\top)^{-1} & l < i < n, \end{cases}$$

and

$$\mathbf{F} = \mathbf{F}_{l+1}^\top \dots \mathbf{F}_{n-1}^\top,$$

where $\mathbf{F}_n$ is the Jacobian of the state transition function f at time n, and $\boldsymbol{\Sigma}_{ii}$ is the i-th element of the block vector $\mathbf{D}_n$. Therefore, the new matrices $\boldsymbol{\Phi}_i$ are computed in constant time by copying the corresponding block of $\mathbf{D}_n$. $\mathbf{F}$ can also be updated in constant time and, the new element $\mathbf{L}_i$ is easily computed using $\mathbf{F}$ at time i and $\mathbf{F}_i$. With this factorization $\mathbf{T}_n$ is not needed to be stored, since by book-keeping $\boldsymbol{\Phi}_i$, $\mathbf{L}_i$, and $\mathbf{F}$, any block of $\mathbf{T}_n$ can be computed in constant time anytime required by the data association process. [1]

This constant time open loop update scheme prompts the necessity to perform data association in times better than linear. This can be done, for instance, in logarithmic time per iteration using a tree structures [14, 18] or even in constant time using grid techniques when covariances are bounded [19].

5. Extension to Hierarchical Mapping

A common approach to reduce the computational cost of SLAM is to resort to hierarchical mapping [19–24]. The idea is to build local maps that are integrated into a global map. Local maps are limited to a bounded number of poses/landmarks and thus, their estimation can be carried out in constant time. Consequently, hierarchical mapping shifts the complexity to the map joining phase. Seminal hierarchical SLAM approaches relied on EKF's both to build the local maps and to assemble the global one [20]. In the long run, the approach is affected by the same computational limitations as EKF-based SLAM. The amortized execution time of this approach can be alleviated by introducing more levels in the hierarchy of maps [19], but memory usage remains the same. More recently, the map joining stage has been addressed using the information-based formulation [24, 25]. In this way, the global map is sparse with the consequent savings in memory usage. The drawback of using the information filter to representations the global map is that the recovery of the global mean and covariances, needed for submap matching, are computationally expensive. The solution used in [24, 25] was to recover the mean and only the necessary columns of the covariance matrix solving a reduced number of separate linear systems.

Next, we show how the information-based hierarchical mapping approaches can benefit from the proposed mixed Kalman-information filter. The advantage of using our approach with respect to Kalman-based map joining strategy is that we can overcome the quadratic memory requirements imposed by the EKF. Conversely, the advantage of the presented approach with respect to information-based map joining is that we obtain a more efficient state recovery.

Similar to [24] and [25], we assume local maps built using a standard EKF. However, our representation of the global map includes only the mean, the information matrix, and the block diagonal and block last column of the covariance matrix of the local maps. The difference with Pose SLAM is that here, blocks correspond to submaps rather than poses. In other words, Pose SLAM can be seen as a fine granularity hierarchical SLAM where each local map includes only one robot pose. From this point of view our approach is related to the continuous submapping strategy suggested in [27].

State augmentation at the upper layer of the hierarchy amounts to adding a locally built map u_n to the global map with the appropriate coordinate shift about the robot pose in x_{n-1},

$$x_n = f(x_{n-1}, u_n). \qquad (13)$$

In Pose SLAM, the measurements link the current pose to previous poses forming loops. In hierarchical mapping, measurement updates register the last local map with previous submaps. The registration is performed by relating shared landmarks between submaps.

[1] The factorization presented here is equivalent to that in [17] but with a slight change in notation that facilitates its generalization to hierarchical SLAM.

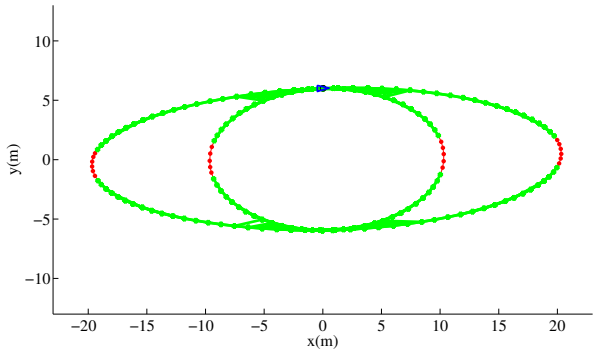


Figure 1: Simulated trajectory when closing all possible loops. The links generated from odometry are shown in red and the links forming loops are overlaid in green.

Given that these are represented in global coordinates in $\mathbf{x}_n$, if a landmark l_n in the current submap is the same as a landmark l_i in any previous submap i, one can define the following measurement function

$$h(\mathbf{x}_n) = l_i - l_n + v_n = 0,$$

with $v_n \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_y)$ the noise inherent to the observations. Then,

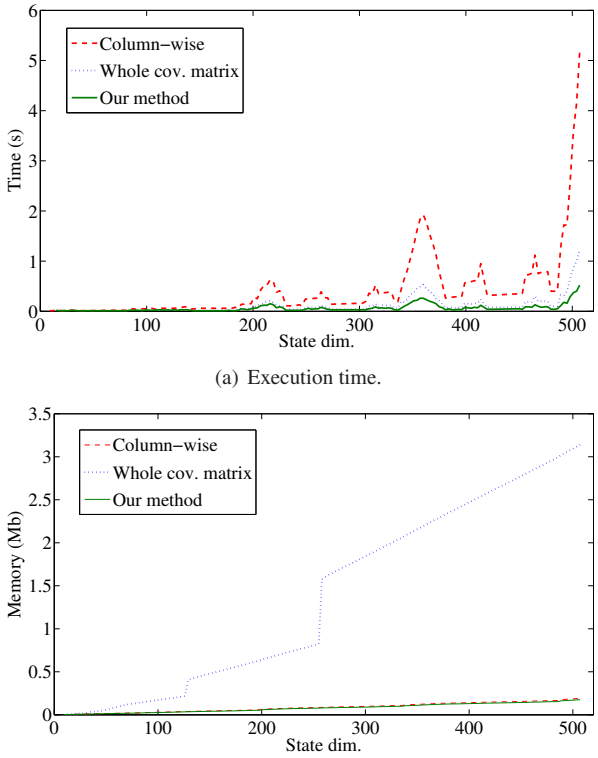
$$\mathbf{H} = [\mathbf{0} \ldots \mathbf{0}\, \mathbf{I}\, \mathbf{0} \ldots \mathbf{0}\, -\mathbf{I}\, \mathbf{0} \ldots \mathbf{0}],$$

where $\mathbf{I}$ is the identity matrix of the adequate size, placed at the positions corresponding to landmarks l_i and l_n.

With this, the state update in Section 3 straightforwardly generalizes to hierarchical mapping, with $\mathbf{T}_i$ and $\mathbf{T}_n$ the block columns of the covariance matrix corresponding to submaps i and n, respectively. Notice that all pairs of shared landmarks between two submaps can be considered simultaneously by defining a measurement function with as many outputs as the number of paired landmarks. In any case, the number of paired landmarks is always below the maximum number of landmarks per submap, which is constant. Thus, considering all paired landmarks at a time only increases the cost by a constant factor.

It is often the case, in hierarchical SLAM, that many loops are formed inside local maps but few are formed between maps. Thus, the robot basically operates in open loop, and we can take advantage of a constant time open loop state recovery strategy equivalent to that described in Section 4. In this case, the last block column of the covariance matrix, $\mathbf{T}_n$, contains the cross covariances $\mathbf{\Sigma}_{in}$ between the current map x_n and any previous map x_i. These block cross covariances can be factorized as

$$\mathbf{\Sigma}_{in} = \mathbf{\Phi}_i \begin{bmatrix} \mathbf{L}_i\, \mathbf{F}\, \mathbf{F}_n^\top & \mathbf{L}_i\, \mathbf{F}\, \mathbf{G} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \qquad (14)$$



(a) Execution time.

(b) Memory footprint.

Figure 2: Execution time and memory footprint for different state recovery strategies when closing a loop in the simulated experiment.

where $\mathbf{0}$ is used to denote zero matrices with the adequate size, $\mathbf{\Phi}_i$ is defined analogously to that in Eq. (12), but on submaps instead of poses. $\mathbf{L}_i$ and $\mathbf{F}$ are the same as those in Section 4, and $\mathbf{G}$ is a block row of the landmark Jacobians of f in Eq. (13) with respect to the robot pose in submap x_{n-1}

$$\mathbf{G} = \left[\frac{\partial f_{l_1}(x_{n-1}, u_n)}{\partial r_{n-1}} \cdots \frac{\partial f_{l_m}(x_{n-1}, u_n)}{\partial r_{n-1}} \right],$$

where r_{n-1} is given in the global frame. Observe that, when no landmarks are present, this cross-covariance factorization reduces to that of Pose SLAM. The structure of the matrix used to compute $\mathbf{\Sigma}_{in}$ from $\mathbf{\Phi}_i$ in Eq. (14) indicates that, in open loop, the new submaps are only related to the previous submaps through the chain of poses from the last loop closure to the new submap.

6. Experiments and Results

This section describes experiments that validate the presented mixed Kalman-information filtering ap-

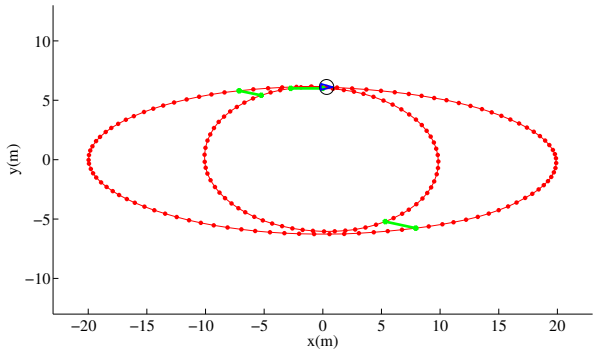


Figure 3: Simulated trajectory when carefully selecting the loops to close using information-based criteria. The links generated from odometry are shown in red and the loop closure links in green.

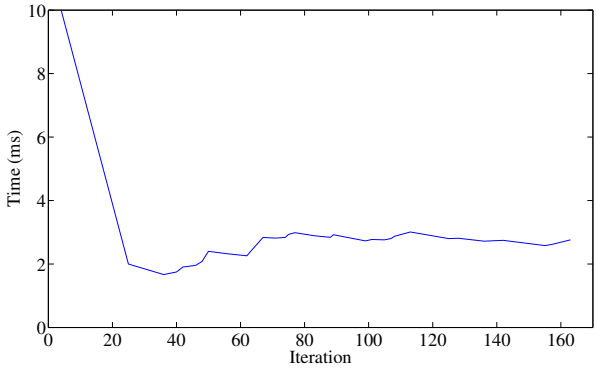


Figure 4: Amortized execution time when controlling the number of loop closures in the simulated experiment.

proach, first using synthetic data and then using real data sets.

In the first experiment, we simulate a robot moving about 0.8 m per step looping around two concentric ellipses, the first with semi-axes 10 m and 6 m and the second with semi-axes 20 m and 6 m. In the simulation, the motion of the robot is measured with an odometric sensor whose error is 5% of the displacement in x and y, and 0.017 rad in orientation. A second sensor is able to establish a link between any two poses closer than ± 3 m in x and y, and ± 0.26 rad in orientation, respectively. This sensor has a noise covariance of $\mathbf{\Sigma}_y = \text{diag}(0.2\,\text{m}, 0.2\,\text{m}, 0.009\,\text{rad})^2$. The simulation is implemented in Matlab running under Linux on a Intel Core 2 at 2.4 GHz with 2 GB of memory.

Fig. 1 shows the estimated trajectory when incorporating all possible loop closure links. We compare the loop closure state update proposed in this paper with the two alternative methods described in Section 2.2. Fig. 2 shows the execution time and the memory footprint for the three approaches. The blue dotted-lines depict the time and memory requirements when recovering the whole covariance matrix, $\mathbf{\Sigma}_n$, as a function of the size of the state at iteration n. The red dashed-lines show the time and memory requirements for the strategy which recovers each block column of the covariance matrix solving one linear system at a time. The results corresponding to the method introduced in this paper are shown in green. In all cases, linear systems are solved using supernodal sparse Cholesky factorization [16] as implemented in [28]. Due to the extra cost of defining the different linear systems to be solved, the time needed to solve separate systems per block column is bigger than that of recovering the whole covariance matrix. However, the memory requirements to solve the whole covariance matrix increase much faster than when solving the systems column-wise. In contrast, the execution time and memory usage of our strategy outperforms the two other methods in both aspects, time and memory usage.

The peaks in the execution time for the three approaches in Fig. 2(a) correspond to poses where many loops are closed. When carefully selecting the loops to be closed using an information-based criterion [12, 14], the robot operates most of the time in open loop. Thus, we can take advantage of the factorization proposed in Section 4. Fig. 3 shows the result of a simulation of the same experiment as that in Fig. 1 when using this strategy. Fig. 4 shows the amortized cost, c_i, at each iteration i, computed as

$$c_i = \frac{1}{i} \sum_{k=1}^{i} t_k, \tag{15}$$

where t_k includes the time for filter related operations at iteration k (the time to compute $\boldsymbol{\mu}$, $\mathbf{D}$, $\mathbf{\Phi}$, and $\mathbf{\Lambda}$, both in open loop and when closing loops), disregarding the cost of sensor registration. After an initial transitory, the plot indicates that the amortized cost is nearly constant for the entire experiment.

To test the performance of the proposed approach in larger problems, we use the simulated Manhattan data set [10] including 10000 poses in $SE(2)$. For this experiment we set $\mathbf{\Sigma}_u = \mathbf{\Sigma}_y = \text{diag}(0.05\,\text{m}, 0.05\,\text{m}, 0.03\,\text{rad})^2$ Fig. 5 shows the final trajectory for this experiment, when considering only the most informative loop closure links. In Fig. 6 we show the execution time and memory footprint for this experiment using three state recovery strategies. The plot for the strategy that recovers the whole $\mathbf{\Sigma}$ stops when the state dimension is about 8000 because Matlab runs out of memory. This clearly indicates that the method that recovers the whole covariance matrix is too memory demanding to be applied to large mapping problems. Both the column-wise

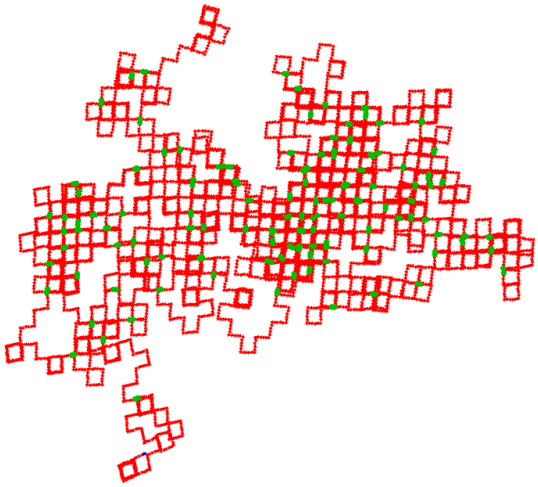
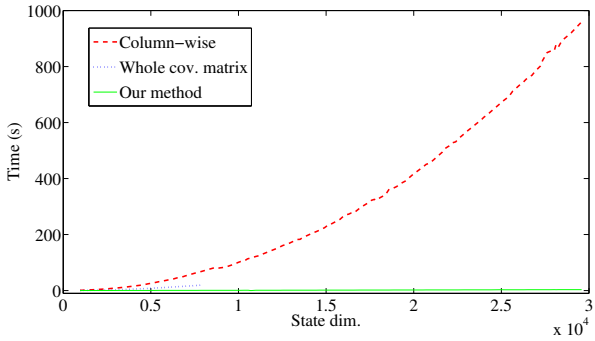


Figure 5: Trajectory in the simulated Manhattan world. The links generated from odometry are shown in red and the informative links forming loops are shown in green.

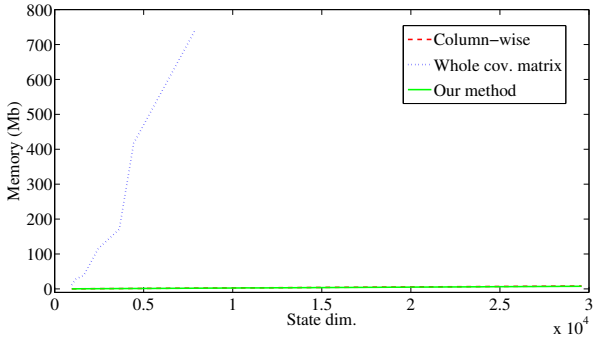
and our strategy are much less demanding with respect to memory use. However, as shown in Fig. 6(a), the column-wise strategy is extremely demanding with respect to computational time while our strategy is not. Note that this simulation is intentionally loopy and that, despite considering only the most informative loops the cost using our strategy scales linearly, even when amortized. As we show next, this situation is not likely to happen in real experiments.

To test the performance of the proposed approach in real situations, we used the Intel data set from [29]. This data set includes 13631 laser scans and the corresponding odometry readings. The laser scans are used to generate sensor-based odometry and to assert loop closures aligning them using a scan matching algorithm [30]. Robot odometry and laser scan matching are modeled with noise covariances $\mathbf{\Sigma}_u = \text{diag}(0.05\,\text{m}, 0.05\,\text{m}, 0.03\,\text{rad})^2$ and $\mathbf{\Sigma}_y = \text{diag}(0.05\,\text{m}, 0.05\,\text{m}, 0.009\,\text{rad})^2$, respectively. Finally, the covariance of the initial pose [2] is set to $\mathbf{\Sigma}_0 = \text{diag}(0.1\,\text{m}, 0.1\,\text{m}, 0.09\,\text{rad})^2$. Due to its large size, this data set is typically pre-processed and reduced to about 1000 poses with about 3500 loop closure links [5]. By carefully selecting the most informative loops [12] we are able to reduce it to only about 100 links, without exposing map accuracy. Fig. 7 shows the final estimated

[2]The main contribution of the paper is reduced computational cost. This cost is given as a complexity bound which is not jeopardized by the selection of values for Σ_0 or any other initial parameter. We give these parameter values explicitly throughout the text only to ease replicability of results.



(a) Execution time.



(b) Memory footprint.

Figure 6: Execution time and memory footprint for different state recovery strategies when closing a loop in the Manhattan experiment.

trajectory.

Fig. 8 shows the execution time and memory footprint at each step using the three different state recovery strategies discussed in the simulated examples. The results confirm that for larger SLAM problems, our method clearly outperforms the two other methods both in memory usage and in execution time.

Fig. 9 shows the amortized time for the whole execution on the Intel experiment. The amortized cost for the state estimation process, but without considering sensor registration, is almost constant. Sensor registration can be carried on in logarithmic time [14] and, with this, the total amortized cost of the presented SLAM system would be logarithmic.

We also applied our state recovery strategy to the Victoria Park data set, a standard data set often used to test hierarchical SLAM algorithms [19, 23]. This data set includes about 6900 laser scans with the corresponding odometry readings. The laser scans are processed to detect the trunks of the trees in the park that are used as landmarks. The hierarchical SLAM state recovery method described in Section 5 was implemented with initial conditions $\mathbf{\Sigma}_0 = \text{diag}(0.25\,\text{m}, 0.25\,\text{m}, 0.017\,\text{rad})^2$, and noise

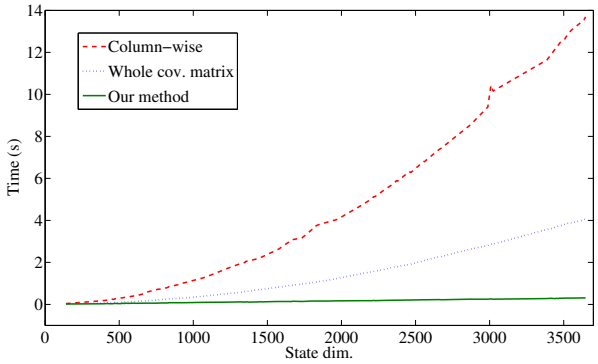
Figure 7: Filtered trajectory using encoder and laser odometry of the Intel data set. The blue arrow indicates the final pose of the robot and the black ellipse the associated covariance at a 95% confidence level.

parameters $\mathbf{\Sigma}_u = \text{diag}(0.025\,\text{m}, 0.025\,\text{m}, 0.017\,\text{m})^2$, and $\mathbf{\Sigma}_y = \text{diag}(0.1\,\text{m}, 0.1\,\text{m})^2$. Local map sizes were arbitrarily limited to 20 landmarks each.

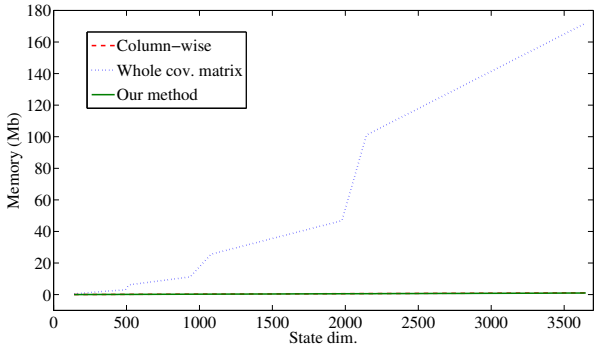
Fig. 10 shows the final trajectory estimate together with the detected landmarks. At the end of the execution, the global map includes 29 submaps. The rectangles in the Fig. 10 bound the area covered by five of these submaps. Note that since there is substantial overlap among submaps, the number of landmarks inside a bounding box might be larger than 20. The trajectory re-traverses many times the same areas and the final graph of submaps includes 113 loop closure links between submaps. Note that, in principle we could form loop closure links when registering consecutive submaps. In that case, the open loop state recovery strategy described in Section 5 would be hardly applicable. To avoid this, we anchor each new submap using the landmarks from the previous submap rotated and translated to the new local reference frame. This only increases the cost of building a submap by a constant factor and the result is a submap that is already properly registered with respect to the previous map.

Fig. 11 shows the execution time and the memory use for the whole execution of the Victoria Park data set for the same three state recovery strategies analyzes before. Our strategy clearly outperforms the other two strategies both in memory use and in execution time.

The maps joining information matrix is rather sparse. The 113 loop closure links amount to less than 15% of the possible links between the 29 submaps produced.



(a) Execution time.



(b) Memory footprint.

Figure 8: Execution time and memory footprint for different state recovery strategies when closing a loop in the Intel experiment.

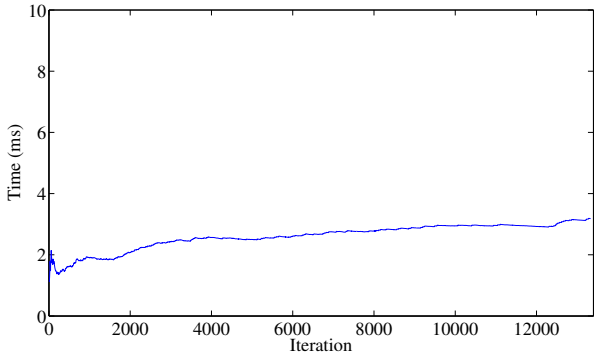


Figure 9: Amortized execution time when controlling the number of loop closures in the Intel experiment.

Since submaps only share a small amount of landmarks, at the lowest level in the hierarchy, the information matrix sparsity is even larger, including only 4% non-zero entries. As shown in Fig. 12, the cost of state recovery after closing loops is amortized over the periods where the robot operates in open loop, and when the local maps can be joined in constant time. The final result is that the state is updated in amortized constant

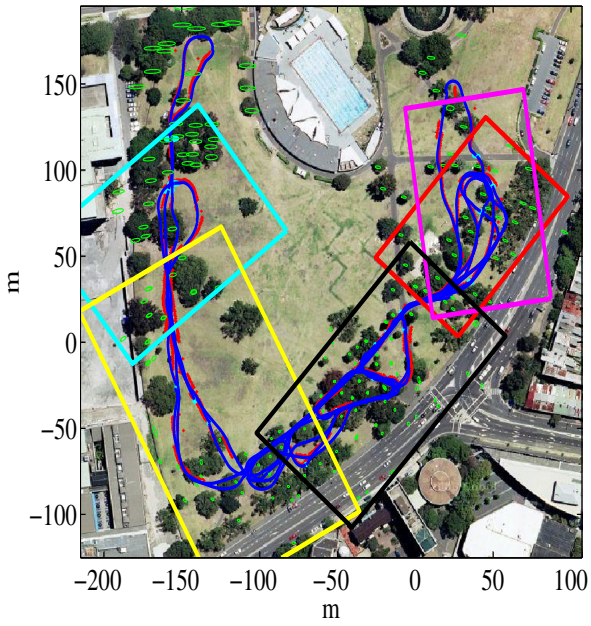
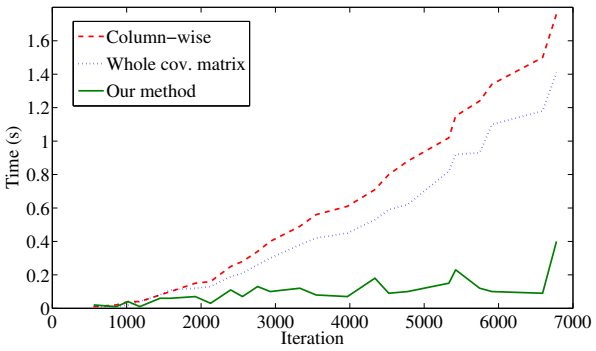
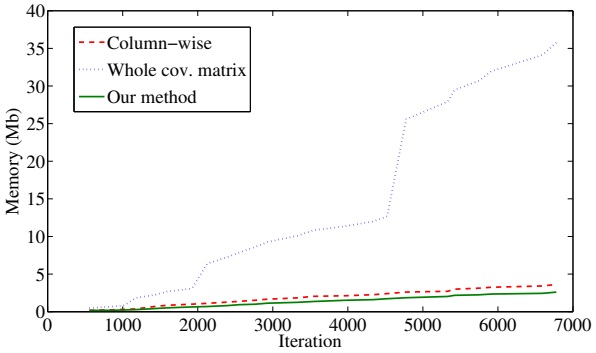


Figure 10: Hierarchical mapping of the Victoria Park data set. The rectangles bound the landmarks included in five of the submaps. The blue line is the estimated trajectory, the red line under the trajectory corresponds to the GPS ground truth, and the green ellipses indicate estimated landmarks and their covariances.



(a) Execution time.



(b) Memory footprint.

Figure 11: Execution time and memory footprint for different state recovery strategies when closing a loop in the Victoria Park experiment.

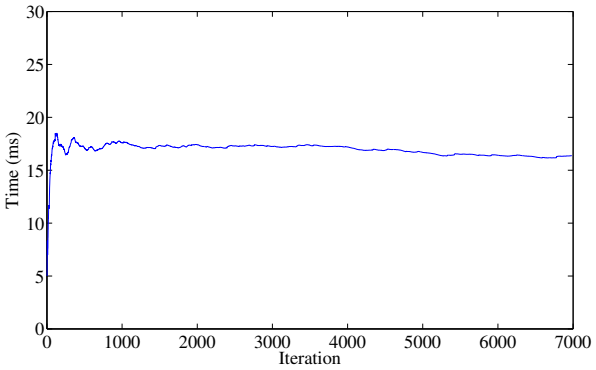


Figure 12: Amortized execution time for global map management in the Victoria Park experiment.

time for local map management and their integration in the global map, but disregarding the cost of sensor registration. The amortized cost for this experiment is higher than for the other two data sets due to the fact that here, all basic map management operations are performed over submaps instead of just on a single pose.

7. Conclusions

The problem of estimating a set of reference frames with relative constraints between them is a fundamental problem in SLAM. It appears, for instance, in Pose SLAM where reference frames are attached to each one of the poses along the robot trajectory or in hierarchical SLAM where reference frames are attached to each local map. When assuming Gaussian distributions, the Kalman and the information filters are the two alternative filtering schemes that have been applied to this problem. In the Kalman filter the mean and the covariance are directly available for linearization and data association, but with quadratic memory cost and time complexity. The information filter offers linear memory cost, but the mean and the covariance need to be recovered from the information vector and the information matrix by solving large linear systems, a process that is computationally too expensive for large problems. Sparse linear algebra tools alleviate either computational time, by recovering the whole covariance matrix, or the memory footprint, by recovering the covariance matrix column-wise, but not both.

In this paper we proposed a mixed Kalman-information approach which maintains the state mean, the block diagonal and block last column entries of

the covariance matrix and the information matrix. The mean and the covariance entries are used to linearize the system when necessary and to perform data association, while the information matrix stores in a very compact way the whole set of correlations between poses. The result is an estimation mechanism that scales linearly both in memory and execution time.

Moreover, both in Pose and in hierarchical SLAM, it is typical to operate most of the time in open loop while exploring new areas or when defining new submaps, as well as to establish only few constraints between the current robot pose (or current submap) and previous poses (or submaps). We have shown that this particular property can be exploited to derive a system whose amortized cost per step is constant. The presented results using simulated experiments and standard SLAM data sets validate the approach.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Innovation under the *"Programa Nacional de Movilidad de Recursos Humanos de Investigación"* to V. Ila and the projects DPI-2007-60858, DPI-2008-06022, MIPRCV Consolider-Ingenio 2010, and the EU URUS project IST-FP6-STREP-045062.

References

[1] R. C. Smith, P. Cheeseman, On the representation and estimation of spatial uncertainty, Int. J. Robot. Res. 5 (4) (1986) 56–68.

[2] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, H. Durrant-Whyte, Simultaneous localization and mapping with sparse extended information filters, Int. J. Robot. Res. 23 (7-8) (2004) 693–716.

[3] M. Montemerlo, S. Thrun, FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics, Vol. 27 of Springer Tracts in Advanced Robotics, Springer, 2007.

[4] F. Dellaert, M. Kaess, Square root SAM: Simultaneous localization and mapping via square root information smoothing, Int. J. Robot. Res. 25 (12) (2006) 1181–1204.

[5] M. Kaess, A. Ranganathan, F. Dellaert, iSAM: Incremental smoothing and mapping, IEEE Trans. Robot. 24 (6) (2008) 1365–1378.

[6] M. R. Walter, R. M. Eustice, J. J. Leonard, Exactly sparse extended information filters for feature-based SLAM, Int. J. Robot. Res. 26 (4) (2007) 335–359.

[7] Z. Wang, S. Huang, G. Dissanayake, D-SLAM: A decoupled solution to simultaneous localization and mapping, Int. J. Robot. Res. 26 (2) (2007) 187–204.

[8] R. M. Eustice, H. Singh, J. J. Leonard, Exactly sparse delayed-state filters for view-based SLAM, IEEE Trans. Robot. 22 (6) (2006) 1100–1114.

[9] K. Konolige, M. Agrawal, FrameSLAM: from bundle adjustment to realtime visual mapping, IEEE Trans. Robot. 24 (5) (2008) 1066–1077.

[10] E. Olson, J. Leonard, S. Teller, Fast iterative alignment of pose graphs with poor initial estimates, in: Proc. IEEE Int. Conf. Robot. Automat., Orlando, 2006, pp. 2262–2269.

[11] G. Grisetti, C. Stachniss, S. Grzonka, W. Burgard, A tree parameterization for efficiently computing maximum likelihood maps using gradient descent, in: Robotics: Science and Systems III, Atlanta, 2007, pp. 9:1–9:8.

[12] V. Ila, J. Andrade-Cetto, R. Valencia, A. Sanfeliu, Vision-based loop closing for delayed state robot mapping, in: Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., San Diego, 2007, pp. 3892–3897.

[13] M. Cummins, P. Newman, FAB-MAP: Probabilistic localization and mapping in the space of appearance, Int. J. Robot. Res. 27 (6) (2008) 647–665.

[14] V. Ila, J. M. Porta, J. Andrade-Cetto, Information-based compact Pose SLAM, IEEE Trans. Robot. 26 (1) (2010) 78–93.

[15] L. M. Paz, P. Piniés, J. D. Tardós, J. Neira, Large scale 6dof SLAM with stereo-in-hand, IEEE Trans. Robot. 24 (5) (2008) 946–957.

[16] Y. Chen, T. A. Davis, W. W. Hager, S. Rajamanickam, Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate, ACM T. Math. Software 35 (3) (2008) 22:1–22:14.

[17] V. Ila, J. M. Porta, J. Andrade-Cetto, Amortized constant time state estimation in SLAM using a mixed Kalman-information filter, in: Proc. European Conf. Mobile Robotics, Dubrovnik, 2009, pp. 211–216.

[18] J. Uhlmann, Introduction to the algorithmics of data association in multiple-target tracking, in: M. E. Liggins, D. E. Hall, J. Llinas (Eds.), Handbook of Multisensor Data Fusion, 2nd Edition, CRC Press, Boca Raton, 2009, Ch. 4, pp. 69–87.

[19] L. M. Paz, J. D. Tardós, J. Neira, Divide and conquer: EKF SLAM in $O(n)$, IEEE Trans. Robot. 24 (5) (2008) 1107–1120.

[20] J. D. Tardós, J. Neira, P. M. Newman, J. J. Leonard, Robust mapping and localization in indoor environments using sonar data, Int. J. Robot. Res. 21 (4) (2002) 311–330.

[21] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, S. Teller, An *atlas* framework for scalable mapping, in: Proc. IEEE Int. Conf. Robot. Automat., Taipei, 2003, pp. 1899–1906.

[22] C. Estrada, J. Neira, J. Tardós, Hierarchical SLAM: Real-time accurate mapping of large environments, IEEE Trans. Robot. 21 (4) (2005) 588–596.

[23] N. Kai, D. Steedly, F. Dellaert, Tectonic SAM: Exact, out-of-core, submap-based SLAM, in: Proc. IEEE Int. Conf. Robot. Automat., Barcelona, 2005, pp. 1678–1685.

[24] S. Huang, Z. Wang, G. Dissanayake, Sparse local submap joining filter for building large-scale maps, IEEE Trans. Robot. 24 (5) (2008) 1121–1130.

[25] C. Cadena, F. Ramos, J. Neira, Efficient large scale SLAM including data association using the combined filter, in: Proc. European Conf. Mobile Robotics, Dubrovnik, 2009, pp. 217–222.

[26] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, M. Csorba, A solution to the simultaneous localization and map building (SLAM) problem, IEEE Trans. Robot. Automat. 17 (3) (2001) 229–241.

[27] G. Sibley, C. Mei, I. Reid, P. Newman, Adaptive relative bundle adjustment, in: Robotics: Science and Systems V, Seattle, USA, 2009, pp. 23:1–23:8.

[28] T. Davis, The SuiteSparse (version 3.4), http://www.cise.ufl.edu/research/sparse/SuiteSparse (2009).

[29] A. Howard, N. Roy, The robotics data set repository (Radish), http://radish.sourceforge.net (2003).

[30] F. Lu, E. Milios, Globally consistent range scan alignment for environment mapping, Auton. Robot. 4 (4) (1997) 333–349.