Power-SLAM: A Linear-Complexity, Anytime Algorithm for SLAM

Esha D. Nerurkar, *Student Member, IEEE,* Stergios I. Roumeliotis, *Member, IEEE*

***Abstract*—In this paper, we present an Extended Kalman Filter (EKF)-based estimator for simultaneous localization and mapping (SLAM) with processing requirements that are linear in the number of features in the map. The proposed algorithm, called the Power SLAM, is based on three key ideas. Firstly, by introducing the Global Map Postponement method, approximations necessary for ensuring linear computational complexity of EKF-based SLAM are delayed over multiple time steps. Then by employing the Power Method, only the most informative of the Kalman vectors, generated during the postponement phase, are retained for updating the covariance matrix. This ensures that the information loss during each approximation epoch is minimized. Next, linear-complexity, rank-2 updates, that minimize the trace of the covariance matrix, are employed to increase the speed of convergence of the estimator. The resulting estimator, in addition to being conservative as compared to the standard EKF, has processing requirements that can be adjusted to the availability of computational resources. Simulation and experimental results are presented that demonstrate the accuracy of the proposed algorithm (Power-SLAM) when compared to the quadratic computational cost standard EKF-based SLAM, and two linear-complexity competing alternatives.**

***Index Terms*—Simultaneous Localization and Mapping, Power Method, Global Map Postponement**

I. Introduction

ONE of the most challenging problems faced in autonomous navigation is Simultaneous Localization and Mapping (SLAM), where robots jointly estimate their own pose (i.e., position and orientation) and model the environment. Mobile robot tasks such as search and rescue missions, and space and underwater exploration are classical examples of SLAM, where (i) the robots do not have access to global positioning devices, such as GPS, or information from such sources is unreliable (e.g., in urban environments or underwater), and (ii) *a priori* information about the environment (e.g., a map) is not available to the robot.

SLAM has been studied extensively in the literature and numerous solutions have been proposed. These solutions differ primarily in the assumptions made for the environment (static or dynamic), the map representation (point/line/plane features, global or robocentric mapping, etc.), the robots' sensors (laser scanners, cameras, etc.) and the estimation framework used (Extended Kalman Filter, Particle Filter, Maximum A Posteriori Estimator, etc.). Amongst these, probably the most commonly used estimator for SLAM is the Extended Kalman Filter (EKF), due to its ease of implementation. Additionally, the EKF is optimal, up to linearization errors, in the Minimum Mean Square Error (MMSE) sense and it recursively computes a concrete measure of the *uncertainty* in the state estimates, namely the covariance matrix. This covariance matrix provides crucial information necessary for minimizing the risk of failure while making decisions related to data association and path planning.

Unfortunately, storing and updating this covariance matrix in EKF-based SLAM is a major bottleneck. Even under the assumption that only few map features are detected at each time step, both the memory and computational requirements of EKF-based SLAM are quadratic, $O(N^2)$, in the number of features, N, in the map. While storage requirements can be handled efficiently by the memory devices available today, the computational complexity has prevented the deployment of mobile robots in large-scale environments. Another critical drawback of using the EKF estimator, due to the non-linear nature of the SLAM problem, is its inherent inconsistency[1] *over time*. Estimator consistency is vital for SLAM because an inconsistent estimator provides no guarantee for the accuracy of the generated state estimates, hence rendering the robot/landmark estimates unreliable.

As detailed in the following section, a number of EKF-based approaches exist that address the computational complexity of SLAM by: (i) delaying the quadratic covariance update step or (ii) employing an approximate structure for the estimator. The main limitation of the methods under the first category, is that inevitably at some point the delayed covariance update will have to be carried out, incurring a computational cost of $O(N^2)$. For large values of N, this can become prohibitive. On the other hand, many of the approximate approaches do not maintain the cross-correlations between the robot's and the landmarks' estimates, which can lead to inconsistency and divergence of the EKF. Furthermore, amongst the approximate approaches that do maintain these correlations, information is discarded during *every* time step, often based on criteria that do *not* guarantee the best use of the available CPU cycles, thus resulting into suboptimal estimators.

To address this problem, in this paper we describe an EKF-based algorithm for SLAM with *linear* computational complexity in the number of features in the map. The proposed conservative[2] approximate estimator *minimizes the information discarded* over *multiple time steps*. This is achieved by:

E. D. Nerurkar, and S. I. Roumeliotis are with the Department of Computer Science and Engineering, Univ. of Minnesota USA e-mail: {nerurkar,stergios}@cs.umn.edu.

[1] A state estimator is consistent if its estimation errors are zero-mean and have covariance matrix smaller or equal to the one calculated by the filter [Bar-Shalom et al., 2001].

[2] An approximate EKF-based SLAM estimator is conservative if its estimated covariance is larger than that of the corresponding standard EKF.

(i) extending the time horizon over which approximations are invoked by using the Global Map Postponement technique [cf. Section III-B], and (ii) using the Power method to compute and retain, after each approximation, only the *most informative* updates (i.e., Kalman vectors whose outer product minimizes the trace of the covariance matrix) [cf. Section III-C]. Finally, in order to speed up the rate of convergence of the proposed estimator, rank-2 covariance updates, that minimize the trace of the covariance matrix under the linear computational complexity constraint, are applied at every time step [cf. Section III-D]. The proposed approach is flexible in the sense that the parameters involved at each stage of the algorithm can be adjusted to meet the availability of computational resources. Before presenting the details of the Power-SLAM algorithm, we briefly review some of the representative EKF-based SLAM approaches.

II. Related Work

In their seminal work, Smith, Self and Cheeseman [Smith et al., 1990] introduced the concept of the stochastic map and proved that the features' and robot's estimates are not independent, as was previously assumed. By using an EKF-based estimator for solving the SLAM problem, Moutarlier and Chatila [Moutarlier and Chatila, 1989] showed that the complete covariance matrix for both the robot and the features must be maintained in order to ensure consistency of the EKF filter. Reducing the computational burden for real-time application of SLAM has been studied by numerous researchers. The EKF-based solutions can classified into 2 main categories:[3]

A. Optimal EKF-based SLAM

By restructuring the EKF equations, Davison [Davison, 1998] showed that it is possible to process multiple observations of the *same* landmark (map feature) in constant time, while delaying the complete update of the covariance matrix. Knight *et al.* [Knight et al., 2001] extended this idea to the case of sub-maps. As in [Davison, 1998], covariance and state updates are limited to the sub-map (i.e., constant time complexity) until the robot moves outside that particular area. When this happens, the whole map needs to be updated which requires $O(N^2)$ operations. Similarly in [Williams et al., 2002], new sub-maps, each with p features, are initialized at various locations along the trajectory. As long as $p \ll N$, where N is the total number of features in the global map, each sub-map can be updated in constant time (i.e., quadratic, $O(p)^2$, in the number of features in this particular sub-map). However, once all the sub-maps are merged, the computational cost again becomes quadratic, $O(N^2)$, in the *total* number of features.

[3]Although numerous approaches exist for reducing the computational complexity of SLAM, e.g., Particle Filter [Montemerlo et al., 2002], thin junction trees [Paskin, 2003], treemaps and multigrids [Frese, 2006], [Frese et al., 2005], square root SAM [Dellaert and Kaess, 2006], etc, we hereafter limit our discussion of related work to approaches based on the Kalman filtering framework. The main reason for this is that EKF-based approaches provide a direct measure of the uncertainty in the robot pose and map estimates by computing their covariance.

B. Approximate EKF-based SLAM

The method described in [Dissanayake et al., 2000] and [Durrant-Whyte et al., 2000] retains the optimal structure of the EKF-based SLAM algorithm but reduces the number of landmarks considered per update step. This is achieved by selecting, based on their covariance, and processing only the most informative features; the remaining features are removed from the state vector. Although this algorithm maintains correlations between the robot and the landmarks and is optimal for the number of landmarks retained in the state vector, it introduces an approximation since not all available map features are processed.

Leonard and Feder [Leonard and Feder, 2000] introduced the concept of multiple overlapping sub-map regions (Decoupled Stochastic Maps), each with its own stochastic map. Their approach scales the EKF-based SLAM algorithm to linear computational complexity. However, there exists no proof for the consistency of this method and it is not possible to estimate the impact of the approximation on the map's uncertainty.

In the relative-map approach presented in [Csorba and Durrant-Whyte, 1997] the relative, instead of the absolute positions of the features, are estimated. By excluding the vehicle pose estimate from the state vector, the covariance matrix takes on a simple block-diagonal structure. Hence, the resulting computational complexity for processing each observation becomes constant time. A drawback of this method is that it does not ensure consistency. The Geometric Projection filter [Newman, 1999] can be used to impose the consistency constraint, however, this increases the computational burden to $O(C^3)$, where C is the number of independent constraints that need to be applied. These constraints have to be imposed every time the robot pose is required. Also, this method lacks a common frame of reference and thus it cannot provide a direct update to the robot pose.

Guivant and Nebot's [Guivant and Nebot, 2001] Compressed EKF (CKF) approach combines the ideas of sub-maps and relative maps. By using sub-maps, this algorithm has complexity $O({N_a}^2)$, where N_a is the number of features in the local map. As in the case of [Davison, 1998], [Knight et al., 2001], it postpones the global update which can be carried out with the complexity of a full SLAM update. While this algorithm, in its optimal form has $O(N^2)$ complexity, an approximation was introduced that involves relative maps and operates in linear time. In this case only a subset of the map features are updated.

Julier and Uhlmann introduced the Covariance Intersection (CI) method [Uhlmann et al., 1997] which does not consider the estimates' correlations. Although this estimator is conservative and its computational requirements scale linearly with the number of features, it has very slow convergence. When partial correlation information is available, the Split CI (SCI) [Julier and Uhlmann, 2001] can be employed. This method works better than CI but does not use the complete correlation information and is still as conservative as CI for the robot estimates.

The Sparse Weight Kalman Filter (SWKF) [Julier, 2001]

approach proposed by Julier relies on the sparsification of the Kalman gain matrix. Based on the observation that most of its elements are significantly smaller as compared to the ones corresponding to the robot pose and the observed landmark, these are set to zero. The resulting approximate algorithm has linear computational complexity but generates very conservative estimates.

In our approach, we first present the Global Map Postponement (GMP) technique that reformulates and extends the postponement method [Davison, 1998], [Knight et al., 2001] to the case of the *global* map. We show that by using the GMP, the computational cost of the *exact* EKF-based SLAM remains *linear* in the number of states, N, as long as the number of delayed updates (or equivalently the number of stored Kalman vectors), m, is significantly smaller than N. However, as the robot moves around in the environment and re-observes landmarks, the number of delayed updates, m, increases. In order to ensure that $m \ll N$ (hence linear computational complexity), we employ a low-rank *approximation* that uses the Power method [Golub and Loan, 1996] for computing the largest eigenvalues and the corresponding eigenvectors in linear time. This technique retains the most informative of the Kalman vectors and allows us to extend the postponement horizon indefinitely. Finally, in order to speed up the convergence of our proposed estimator, linear-cost, rank-2 updates, selected so as to minimize the trace of the covariance matrix, are imposed at every time step. Preliminary work on this topic has been presented in [Nerurkar and Roumeliotis, 2007]. In this paper we further present an in-depth complexity analysis of the proposed algorithm along with real-world experimental validation.

III. Algorithm Description

A. Standard EKF-based SLAM

This section introduces the notation used in this paper and briefly describes the EKF-based SLAM equations in 2D. Note that our proposed approach can be easily extended to 3D. The state vector, $\mathbf{x}_k$, consists of:

$$\mathbf{x}_k = \left[\mathbf{x}_{r_k}^T \ , \mathbf{p}_{1_k}^T \ , \mathbf{p}_{2_k}^T, \ldots, \mathbf{p}_{N_k}^T\right]^T. \tag{1}$$

Here, $\mathbf{x}_{r_k} = [x_{r_k}, y_{r_k}, \phi_{r_k}]^T$, denotes the position and orientation of the robot and $\mathbf{p}_{i_k} = [x_{i_k}, y_{i_k}]^T$ denotes the position of the i^{th} landmark, $i = 1, \ldots, N$, at time-step k. All the above quantities are expressed with respect to a global frame of reference.

The robot is equipped with proprioceptive (odometry) sensors that provide linear, v_{m_k}, and rotational, ω_{m_k}, velocity measurements. The robot's motion model is given by:

$$\mathbf{x}_{r_{k+1}} = \mathbf{f}(\mathbf{x}_{r_k}, \mathbf{u}_k, \mathbf{w}_k), \tag{2}$$

where $\mathbf{f}$ is in general a non-linear function and $\mathbf{u}_k = [v_{m_k}, \omega_{m_k}]^T$ is the control input. The vector $\mathbf{w}_k = [w_{v_k}, w_{\omega_k}]^T$, with covariance $\mathbf{Q}_k$, represents the zero-mean, white Gaussian noise in the linear and angular velocity measurements, respectively. Additionally, the robot is equipped with exteroceptive sensors that allow it to measure the distance and bearing to landmark i. The robot's measurement model is given by:

$$\mathbf{z}_{k+1} = \mathbf{h}(\mathbf{x}_{r_{k+1}}, \mathbf{p}_{i_{k+1}}) + \mathbf{n}_{k+1} \tag{3}$$

with $\mathbf{h} = [d_{k+1}, \theta_{k+1}]^T$, where d_{k+1} and θ_{k+1} are the true distance and bearing from the robot to landmark i and $\mathbf{n}_{k+1} = [n_{d_{k+1}}, n_{\theta_{k+1}}]^T$, is the additive zero-mean white Gaussian measurement noise with covariance $\mathbf{R}_{k+1}$.

1) Propagation: The propagation equations for the robot and landmarks' state estimates are given by:

$$\hat{\mathbf{x}}_{r_{k+1|k}} = \mathbf{f}(\hat{\mathbf{x}}_{r_{k|k}}, \mathbf{u}_k, \mathbf{0}) \tag{4}$$

$$\hat{\mathbf{p}}_{i_{k+1|k}} = \hat{\mathbf{p}}_{i_{k|k}}, \quad i = 1, \ldots, N \tag{5}$$

where $\hat{\mathbf{m}}_{j_{l|p}}$ denotes the estimates of the random vector $\mathbf{m}_j$ at time-step l, given all the measurements up to time-step p. Furthermore, (5) results from the assumption that the landmarks are stationary. The covariance propagation equation is given by:

$$\mathbf{P}_{k+1|k} = \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T, \tag{6}$$

where $\mathbf{P}$ is the symmetric state covariance matrix with the following structure:[4]

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_r\mathbf{x}_r} & \mathbf{P}_{\mathbf{x}_r\mathbf{p}_1} & \cdots & \mathbf{P}_{\mathbf{x}_r\mathbf{p}_N} \\ \mathbf{P}_{\mathbf{p}_1\mathbf{x}_r} & \mathbf{P}_{\mathbf{p}_1\mathbf{p}_1} & \cdots & \mathbf{P}_{\mathbf{p}_1\mathbf{p}_N} \\ \vdots & \vdots & & \vdots \\ \mathbf{P}_{\mathbf{p}_N\mathbf{x}_r} & \mathbf{P}_{\mathbf{p}_N\mathbf{p}_1} & \cdots & \mathbf{P}_{\mathbf{p}_N\mathbf{p}_N} \end{bmatrix} \tag{7}$$

and

$$\mathbf{\Phi}_k = \begin{bmatrix} \mathbf{\Phi}_{rk} & \mathbf{0}_{3\times 2N} \\ \mathbf{0}_{3\times 2N}^T & \mathbf{I}_{2N} \end{bmatrix}, \ \mathbf{G}_k = \begin{bmatrix} \mathbf{G}_{rk} \\ \mathbf{0}_{2N\times 2} \end{bmatrix}. \tag{8}$$

Here,

$$\mathbf{\Phi}_{rk} = \nabla_{\mathbf{x}_r}(\mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, \mathbf{0}))$$
$$\mathbf{G}_{rk} = \nabla_{\mathbf{w}}(\mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, \mathbf{0}))$$

and $\mathbf{I}_{2N}$ is the $2N \times 2N$ identity matrix.

2) Update: The estimates for the robot's distance and bearing measurements to landmark i, at time-step $k+1$, are given by:

$$\hat{\mathbf{z}}_{k+1} = \mathbf{h}(\hat{\mathbf{x}}_{r_{k+1}}, \hat{\mathbf{p}}_{i_{k+1}}) = \mathbf{h}(\hat{\mathbf{x}}_{k+1}). \tag{9}$$

Once the actual landmark measurement, $\mathbf{z}_{k+1}$, is obtained, the state and covariance are updated as follows:

$$\mathbf{r}_{k+1} = \mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1} \tag{10}$$

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1} \mathbf{P}_{k+1|k} \mathbf{H}_{k+1}^T + \mathbf{R}_{k+1} \tag{11}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k} \mathbf{H}_{k+1}^T \mathbf{S}_{k+1}^{-1} \tag{12}$$

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1} \mathbf{r}_{k+1} \tag{13}$$

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{K}_{k+1} \mathbf{S}_{k+1} \mathbf{K}_{k+1}^T \tag{14}$$

where the measurement matrix, $\mathbf{H}_{k+1}$, is given by:

$$\mathbf{H}_{k+1} = [\mathbf{H}_r \ \mathbf{0}_{2\times 2(i-1)} \ \mathbf{H}_i \ \mathbf{0}_{2\times 2(N-i)}] \tag{15}$$

[4] The time subscripts are omitted here to simplify the presentation.

TABLE I
DIMENSIONS OF TERMS APPEARING IN EKF-BASED SLAM

Terms in EKF-based SLAM		Dimension
State vector	$\mathbf{x}_k$	$(2N+3)\times 1$
State Covariance	$\mathbf{P}_k$	$(2N+3)\times(2N+3)$
Jacobian of $\mathbf{f}$ w.r.t. $\mathbf{x}_r$	$\mathbf{\Phi}_{rk}$	3×3
Jacobian of $\mathbf{f}$ w.r.t. $\mathbf{w}$	$\mathbf{G}_{rk}$	3×2
State Transition matrix	$\mathbf{\Phi}_k$	$(2N+3)\times(2N+3)$
Jacobian of $\mathbf{h}$ w.r.t. $\mathbf{x}_r$	$\mathbf{H}_r$	2×3
Jacobian of $\mathbf{h}$ w.r.t. $\mathbf{p}_i$	$\mathbf{H}_i$	2×2
Measurement matrix	$\mathbf{H}_{k+1}$	$2\times(2N+3)$
Measurement Residual	$\mathbf{r}_{k+1}$	2×1
Residual Covariance	$\mathbf{S}_{k+1}$	2×2
Kalman Gain	$\mathbf{K}_{k+1}$	$(2N+3)\times 2$

and $\mathbf{H}_r = \nabla_{\mathbf{x}_r}(\mathbf{h}(\hat{\mathbf{x}}_{k+1|k}))$, $\mathbf{H}_i = \nabla_{\mathbf{p}_i}(\mathbf{h}(\hat{\mathbf{x}}_{k+1|k}))$. Here, the quantities $\mathbf{r}_{k+1}$, $\mathbf{S}_{k+1}$, $\mathbf{K}_{k+1}$, $\hat{\mathbf{x}}_{k+1|k+1}$, and $\mathbf{P}_{k+1|k+1}$ denote the measurement residual vector, the residual covariance matrix, the Kalman gain matrix, the updated state vector, and the updated covariance matrix respectively, at time-step $k+1$. Table I lists the dimensions of the various quantities that appear in the EKF-based SLAM formulation.

The $O(N^2)$ computational complexity of EKF-based SLAM arises due to the covariance update step [cf. (14)], which involves multiplication of the Kalman gain matrix of dimensions $(2N+3)\times 2$ [cf. (12)]. For robots involved in exploratory tasks or mapping of dense environments, N, i.e., the number of landmarks, continually increases, and hence the cost of updating the covariance matrix can prohibit real-time performance.

In order to overcome this computational bottleneck, in the next section we propose the Global Map Postponement (GMP) approach and show that as long as $m \ll N$ (m is the number of delayed updates), the computational complexity of EKF-based SLAM can be reduced from quadratic to linear in N, *without* requiring any approximation.

B. Global Map Postponement SLAM

Contrary to the approaches of [Davison, 1998], [Knight et al., 2001] that support postponement only when the robot operates within small areas (sub-maps), we hereafter present our GMP method which poses no restrictions on the motion of the robot. Our goal in this section is to demonstrate that, within the GMP framework, exact EKF-based SLAM requires only $O(N)$ operations per time step.

Consider the case where at time-step $k+1$, a new landmark observation is processed to update the covariance. In GMP, (14) is reformulated as:

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - (\mathbf{K}_{k+1}\mathbf{S}_{k+1}^{1/2})(\mathbf{K}_{k+1}\mathbf{S}_{k+1}^{1/2})^T, \quad (16)$$

where $\mathbf{S}^{1/2}$ is a lower-triangular matrix obtained from the Cholesky factorization of $\mathbf{S}$. Since $\mathbf{S}$ is a 2×2 matrix, this Cholesky factorization is carried out in constant time. The dimensions of the resulting term, $(\mathbf{K}_{k+1}\mathbf{S}_{k+1}^{1/2})$, are $(2N+3)\times 2$ and this matrix is split into two vectors, $\mathbf{k}_1$, $\mathbf{k}_2$, each of dimensions $(2N+3)\times 1$. This gives us:

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \sum_{i=1}^{2}\mathbf{k}_i\mathbf{k}_i^T, \quad (17)$$

where $\mathbf{k}_i$ is the i^{th} column of $\mathbf{K}_{k+1}\mathbf{S}_{k+1}^{1/2}$ and furthermore $\mathbf{k}_i = \sqrt{\lambda_i}\mathbf{v}_i$, where $\mathbf{v}_i$ is the eigenvector corresponding to the eigenvalue, λ_i, of $\mathbf{K}_{k+1}\mathbf{S}_{k+1}\mathbf{K}_{k+1}^T$. Here, it is important to note that in the GMP approach, the vector outer-product sum, $\sum_{i=1}^{2}\mathbf{k}_i\mathbf{k}_i^T$, is *never* computed. Instead the Kalman vectors, $\mathbf{k}_i$, are stored for later processing[5].

Maintaining this sum of vector outer-products forms the framework of GMP and is directly responsible for reducing the computational complexity of EKF-based SLAM. Therefore for clarity, we divide the discussion on GMP into two parts: (i) first we demonstrate how this structure can be maintained through subsequent propagation and update steps, and then (ii) we present the computational complexity of GMP for propagation, update and landmark initialization.

1) *Propagation*: The covariance propagation equation at the next time step, i.e., time-step $k+2$, is given by [cf. (6), (17)]:

$$\begin{aligned}\mathbf{P}_{k+2|k+1} &= \mathbf{\Phi}_{k+1}\mathbf{P}_{k+1|k}\mathbf{\Phi}_{k+1}^T + \mathbf{G}_{k+1}\mathbf{Q}_{k+1}\mathbf{G}_{k+1}^T \\ &\quad - \sum_{i=1}^{2}(\mathbf{\Phi}_{k+1}\mathbf{k}_i)(\mathbf{\Phi}_{k+1}\mathbf{k}_i)^T \\ &= \mathbf{P}_{k+2|k} - \sum_{i=1}^{2}\mathbf{k}_i^*\mathbf{k}_i^{*T}, \end{aligned} \quad (18)$$

where $\mathbf{k}_i^* = \mathbf{\Phi}_{k+1}\mathbf{k}_i$. At this step, in the GMP approach the quantities $\mathbf{P}_{k+2|k}$ and $\mathbf{k}_i^*$ are evaluated but the vector product, $\mathbf{k}_i^*\mathbf{k}_i^{*T}$, is not computed. Instead the vectors, $\mathbf{k}_i^*$, are stored for later processing.

2) *Update*: The residual covariance is given by [cf. (11), (18)]:

$$\begin{aligned}\mathbf{S}_{k+2} &= \mathbf{H}_{k+2}\mathbf{P}_{k+2|k}\mathbf{H}_{k+2}^T + \mathbf{R}_{k+2} \\ &\quad - \sum_{i=1}^{2}(\mathbf{H}_{k+2}\mathbf{k}_i^*)(\mathbf{H}_{k+2}\mathbf{k}_i^*)^T.\end{aligned}$$

Here, $\mathbf{S}_{k+2}$ is evaluated, i.e., the vector outer-product sum in the above equation is explicitly calculated. Next, the Kalman gain is expressed as [cf. (12), (18)]:

$$\mathbf{K}_{k+2} = \mathbf{P}_{k+2|k}\mathbf{H}_{k+2}^T\mathbf{S}_{k+2}^{-1} - \sum_{i=1}^{2}\mathbf{k}_i^*(\mathbf{H}_{k+2}\mathbf{k}_i^*)^T\mathbf{S}_{k+2}^{-1}.$$

Here again, $\mathbf{K}_{k+2}$ is evaluated completely. Once the Kalman gain matrix is obtained, the state update [cf. (13)] is carried out. Finally, the covariance update is expressed as [cf. (14), (17), (18)]:

$$\begin{aligned}\mathbf{P}_{k+2|k+2} &= \mathbf{P}_{k+2|k} - \sum_{i=1}^{2}\mathbf{k}_i^*\mathbf{k}_i^{*T} - \mathbf{K}_{k+2}\mathbf{S}_{k+2}\mathbf{K}_{k+2}^T \\ &= \mathbf{P}_{k+2|k} - \sum_{i=1}^{2}\mathbf{k}_i^*\mathbf{k}_i^{*T} - \sum_{i=1}^{2}\mathbf{k}_i\mathbf{k}_i{}^T \\ &= \mathbf{P}_{k+2|k} - \sum_{i=1}^{4}\mathbf{k}_i\mathbf{k}_i{}^T. \end{aligned} \quad (19)$$

[5]Throughout this paper, we refer to these vectors as "Kalman" vectors. While this is true for the ones appearing during updates (though they are scaled and rotated), we also extend this definition to describe vectors that result later on from the low-rank approximation and/or after sparsifications.

Note that, in (19), for simplifying the notation, we have set $\mathbf{k}_3 = \mathbf{k}_1$, $\mathbf{k}_4 = \mathbf{k}_2$ and $\mathbf{k}_1 = \mathbf{k}_1^*$, $\mathbf{k}_2 = \mathbf{k}_2^*$.

As evident from (18), (19), the vector outer-product sum structure in the GMP is preserved for subsequent propagation and update steps. Specifically, from (19), we see that after each update step, two additional Kalman vectors are generated. Therefore, repeating this process for all subsequent propagation and updates, at time-step $k+m$, the covariance update equation has the form[6]:

$$\mathbf{P}_{k+m|k+m} = \mathbf{P}_{k+m|k} - \sum_{i=1}^{2m} \mathbf{k}_i \mathbf{k}_i^T. \tag{20}$$

Maintaining this structure and assuming that $m \ll N$, we hereafter present the complexity analysis for GMP SLAM and show that both propagation and update can be performed with computational cost at most $O(mN)$. Furthermore, we show that landmark initialization can also be efficiently carried out in the GMP framework with cost at most $O(mN)$.

1) Propagation: The covariance propagation equation at time-step $k+m+1$ is given by [cf. (6), (20)]:

$$\begin{aligned}\mathbf{P}_{k+m+1|k+m} &= \mathbf{\Phi}_{k+m}\mathbf{P}_{k+m|k}\mathbf{\Phi}_{k+m}^T + \mathbf{G}_{k+m}\mathbf{Q}_{k+m}\mathbf{G}_{k+m}^T \\ &\quad - \sum_{i=1}^{2m}(\mathbf{\Phi}_{k+m}\mathbf{k}_i)(\mathbf{\Phi}_{k+m}\mathbf{k}_i)^T \\ &= \mathbf{P}_{k+m+1|k} - \sum_{i=1}^{2m}\mathbf{k}_i^*\mathbf{k}_i^{*T},\end{aligned} \tag{21}$$

where $\mathbf{P}_{k+m+1|k}$ is the propagated covariance at time-step $k+m+1$ given measurements up to time-step k. As in standard EKF-based SLAM, $\mathbf{P}_{k+m+1|k}$ is computed in linear time. Due to the special block-diagonal structure of the $\mathbf{\Phi}_{k+m}$ matrix [cf. (8)], each $\mathbf{k}_i^*$ can be calculated in constant time. Since $2m$ such Kalman vectors have to be calculated, the overall computational complexity of this step is also constant time, i.e., $O(m)$.

2) Update: The residual covariance is given by [cf. (11), (21)]:

$$\begin{aligned}\mathbf{S}_{k+m+1} &= \mathbf{H}_{k+m+1}\mathbf{P}_{k+m+1|k}\mathbf{H}_{k+m+1}^T + \mathbf{R}_{k+m+1} \\ &\quad - \sum_{i=1}^{2m}(\mathbf{H}_{k+m+1}\mathbf{k}_i^*)(\mathbf{H}_{k+m+1}\mathbf{k}_i^*)^T.\end{aligned} \tag{22}$$

As in standard EKF-SLAM, the term $(\mathbf{H}_{k+m+1}\mathbf{P}_{k+m+1|k}\mathbf{H}_{k+m+1}^T + \mathbf{R}_{k+m+1})$ is calculated in constant time. Since $\mathbf{H}_{k+m+1}$ contains only two non-zero blocks [cf. (15)], each term, $\mathbf{H}_{k+m+1}\mathbf{k}_i^*$, is calculated in constant time resulting in a total additional cost of $O(m)$ for calculating $2m$ such terms. Finally, since the dimensions of $\mathbf{H}_{k+m+1}\mathbf{k}_i^*$ are 2×1, each vector outer-product, $(\mathbf{H}_{k+m+1}\mathbf{k}_i^*)(\mathbf{H}_{k+m+1}\mathbf{k}_i^*)^T$, can be evaluated in constant time and the computational complexity for calculating the sum of $2m$ such matrices is $O(m)$. Thus the overall computational complexity for calculating the residual covariance matrix is also $O(m)$.

[6]For simplicity, we assume one measurement per time step, i.e. one update step at every time step.

The Kalman gain is expressed as [cf. (12), (21)]:

$$\begin{aligned}\mathbf{K}_{k+m+1} &= \mathbf{P}_{k+m+1|k}\mathbf{H}_{k+m+1}^T\mathbf{S}_{k+m+1}^{-1} \\ &\quad - \sum_{i=1}^{2m}\mathbf{k}_i^*(\mathbf{H}_{k+m+1}\mathbf{k}_i^*)^T\mathbf{S}_{k+m+1}^{-1}.\end{aligned} \tag{23}$$

Similar to the standard EKF-based SLAM, the first term, $\mathbf{P}_{k+m+1|k}\mathbf{H}_{k+m+1}^T\mathbf{S}_{k+m+1}^{-1}$, is calculated in $O(N)$ (note that since $\mathbf{S}$ is of dimensions 2×2, $\mathbf{S}^{-1}$ is calculated in constant time). Furthermore, since the terms $\mathbf{H}_{k+m+1}\mathbf{k}_i^*$ have already been calculated [cf. (22)], the cost of computing $\mathbf{k}_i^*(\mathbf{H}_{k+m+1}\mathbf{k}_i^*)^T\mathbf{S}_{k+m+1}^{-1}$ is $O(N)$. Thus the summation term in (23) is evaluated in $O(mN)$, leading to an overall computational complexity of $O(mN)$ for this step. Also, once the Kalman gain $\mathbf{K}_{k+m+1}$ is available, the state update [cf. (13)] is carried out in linear time.

Finally, the covariance update is expressed as [cf. (14), (21)]:

$$\begin{aligned}\mathbf{P}_{k+m+1|k+m+1} &= \mathbf{P}_{k+m+1|k} - \sum_{i=1}^{2m}\mathbf{k}_i^*\mathbf{k}_i^{*T} \\ &\quad - \mathbf{K}_{k+m+1}\mathbf{S}_{k+m+1}\mathbf{K}_{k+m+1}^T \\ &= \mathbf{P}_{k+m+1|k} - \sum_{i=1}^{2m}\mathbf{k}_i^*\mathbf{k}_i^{*T} - \sum_{i=1}^{2}\mathbf{k}_i{\mathbf{k}_i}^T,\end{aligned} \tag{24}$$

where $\mathbf{k}_i = (\mathbf{K}_{k+m+1}\mathbf{S}_{k+m+1}^{1/2})$ is the i^{th} column of $(\mathbf{K}_{k+m+1}\mathbf{S}_{k+m+1}^{1/2})$. Again, to simplify the notation, we denote $\mathbf{k}_{2m+1} = \mathbf{k}_1$, $\mathbf{k}_{2m+2} = \mathbf{k}_2$ and $\mathbf{k}_j = \mathbf{k}_j^*$, $j = 1, \ldots, 2m$, to obtain:

$$\mathbf{P}_{k+m+1|k+m+1} = \mathbf{P}_{k+m+1|k} - \sum_{i=1}^{2(m+1)}\mathbf{k}_i{\mathbf{k}_i}^T. \tag{25}$$

At this step, we do not actually evaluate the sum of the outer-product of the Kalman vectors and hence we only consider the computations required for generating the new Kalman vectors. Since the generation of new Kalman vectors only involves the Cholesky factorization of the 2×2 matrix $\mathbf{S}$, the covariance update step in GMP is constant time.

3) Landmark Initialization: Next we describe how landmark initialization can be efficiently carried out in the GMP framework. Every time a new landmark, $N+1$, is detected, an estimate for this landmark, $\hat{\mathbf{p}}_{N+1}$, has to be appended to the state vector. Also, the covariance matrix $\mathbf{P}_{k+m|k+m}$ [cf. (20)] needs to be appropriately augmented. While the new landmark's initial estimate can be generated as in the standard EKF-based SLAM, the following steps have to be carried out for updating the covariance:

1) Firstly, zeros are appended as the last two additional elements of each $\mathbf{k}_i$ vector.
2) Matrix $\mathbf{P}_{k+m|k}$ [cf. (7)] is augmented to include the block matrices that correspond to:[7]
   a) The new landmark's covariance $\mathbf{P}_{\mathbf{p}_{N+1}\mathbf{p}_{N+1}} = \mathbf{H}_{N+1}^T\left(\mathbf{H}_r\mathbf{P}_{\mathbf{x}_r\mathbf{x}_r}\mathbf{H}_r^T + \mathbf{R}\right)\mathbf{H}_{N+1}$.

[7]Time indices have been omitted to simplify the discussion.

TABLE II
COMPUTATIONAL COMPLEXITY OF THE GMP EKF-BASED SLAM ALG. N: NUMBER OF LANDMARKS IN THE MAP, m: NUMBER OF KALMAN VECTORS IN THE VECTOR OUTER-PRODUCT SUM

Steps in GMP EKF-SLAM	Computational Complexity
State Propagation	$O(1)$
Covariance Propagation	$O(N)$
Residual Covariance	$O(m)$
Kalman Gain	$O(mN)$
State Update	$O(N)$
Covariance Update	$O(1)$
Landmark Initialization	$O(mN)$

b) The new landmark's cross-correlation with the robot $\mathbf{P}_{\mathbf{x}_r\mathbf{p}_{N+1}} = -\mathbf{P}_{\mathbf{x}_r\mathbf{x}_r}\mathbf{H}_r^T\mathbf{H}_{N+1}$.
c) The new landmark's cross-correlation terms with each of the N existing landmarks $\mathbf{P}_{\mathbf{p}_i\mathbf{p}_{N+1}} = -\mathbf{P}_{\mathbf{p}_i\mathbf{x}_r}\mathbf{H}_r^T\mathbf{H}_{N+1},\ i=1\dots N$,

where $\mathbf{H}_r$ and $\mathbf{H}_{N+1}$ are the non-zero blocks of the measurement matrix [cf. (15)] corresponding to the observation of landmark $N+1$.

Although $N+2$ terms, as seen above, need to be determined to update the covariance matrix, the cost of calculating each of them is constant once $\mathbf{P}_{\mathbf{x}_r\mathbf{x}_r}$ and $\mathbf{P}_{\mathbf{p}_i\mathbf{x}_r}$ are retrieved. As shown below, we can obtain $\mathbf{P}_{\mathbf{x}_r\mathbf{x}_r}$ and $\mathbf{P}_{\mathbf{p}_i\mathbf{x}_r}$ [cf. (20)] at a cost of $O(m)$ each (note that $\mathbf{P}_{\mathbf{x}_r\mathbf{x}_r}$ and $\mathbf{P}_{\mathbf{p}_i\mathbf{x}_r}$ are also needed for data association and can be determined by the same process). We obtain the 3×3 sub-matrix $\mathbf{P}_{\mathbf{x}_r\mathbf{x}_r}$ as follows:

$$\mathbf{P}_{\mathbf{x}_r\mathbf{x}_{r\,k+m|k+m}} = \mathbf{P}_{\mathbf{x}_r\mathbf{x}_{r\,k+m|k}} - \sum_{i=1}^{2m}\mathbf{k}_{ri}\mathbf{k}_{ri}^T, \qquad (26)$$

where $\mathbf{k}_{ri}$ denotes the first 3 elements of the vector $\mathbf{k}_i$ that correspond to the robot. Since each $\mathbf{k}_{ri}$ vector has dimensions 3×1, $\mathbf{P}_{\mathbf{x}_r\mathbf{x}_{r\,k+m|k+m}}$ can be evaluated in constant time, i.e., $O(m)$. Similarly, the 2×3 sub-matrix $\mathbf{P}_{\mathbf{p}_j\mathbf{x}_r}$ is evaluated in $O(m)$ as follows:

$$\mathbf{P}_{\mathbf{p}_j\mathbf{x}_{r\,k+m|k+m}} = \mathbf{P}_{\mathbf{p}_j\mathbf{x}_{r\,k+m|k}} - \sum_{i=1}^{2m}\mathbf{k}_{p_ji}\mathbf{k}_{ri}^T, \qquad (27)$$

where $\mathbf{k}_{p_ji}$ denotes the 2 elements of the vector $\mathbf{k}_i$ that correspond to landmark $\mathbf{p}_j$. Subsequently, each new term of the covariance matrix, corresponding to landmark $N+1$, can be evaluated at a cost of $O(m)$. Since $N+2$ such terms need to be calculated, the overall cost for inserting a new landmark in the map is $O(mN)$.

From the preceding presentation, it is evident that by using the GMP technique, we can limit the computational complexity of standard EKF-SLAM to $O(mN)$. Table II summarizes the computational complexity of each step of GMP. At this point, it is important to note that since no approximation has been made up to this stage, the GMP EKF-based SLAM will produce exactly the same estimates as the standard EKF-based SLAM, in *linear* time, for as long as the number of delayed updates, m, is significantly smaller than N. Inevitably, however, as the robot navigates and makes new observations, m will continuously increase. Therefore, in order to maintain the structure of the covariance matrix [cf. (20)] while allowing for linear-time updates, it is necessary to devise a technique whereby the number, m, of $\mathbf{k_i}$ vectors in the vector outer-product sum $\sum_{i=1}^{2m}\mathbf{k}_i\mathbf{k}_i^T$ (right-hand side of (20)) remains upper-bounded by a quantity $M_{max} \ll N$. In the following section, we describe how this is achieved by employing a low-rank approximation of $\sum_{i=1}^{2m}\mathbf{k}_i\mathbf{k}_i^T$, based on the Power Method.

C. Low-Rank Approximation

Once the increasing number of Kalman vectors, m, reaches M_{max}, the Power-SLAM algorithm employs a low-rank approximation of the accumulated, rank-M_{max} matrix $\mathbf{D}$:[8]

$$\mathbf{D} = \sum_{i=1}^{M_{max}}\mathbf{k}_i\mathbf{k}_i^T = \sum_{i=1}^{M_{max}}\lambda_i\mathbf{v}_i\mathbf{v}_i^T, \qquad (28)$$

where $\lambda_1 > \lambda_2 \geq \lambda_3 ... \geq \lambda_{M_{max}}$ are the eigenvalues of $\mathbf{D}$ and $\mathbf{v}_i$, $i=1\dots M_{max}$ are the corresponding eigenvectors. The proposed low-rank approximation of $\mathbf{D}$ retains its M_{min} largest eigenvalue-eigenvector pairs:

$$\mathbf{D} \simeq \sum_{i=1}^{M_{min}}\lambda_i\mathbf{v}_i\mathbf{v}_i^T, = \sum_{i=1}^{M_{min}}\mathbf{k}_i^*\mathbf{k}_i^{*T} = \mathbf{D}^*, \qquad (29)$$

where $M_{min} < M_{max}$ and, $\mathbf{k}_i^* = \sqrt{\lambda_i}\mathbf{v}_i$, are the new Kalman vectors. Here it is important to note that this approximation is *optimal* since it retains the most informative vectors, i.e., the scaled eigenvectors that correspond to the largest eigenvalues of $\mathbf{D}$. The motivation for this low-rank approximation is to ensure that only m, where $M_{min} \leq m \leq M_{max} \ll N$, vectors will be involved in further computations [cf. (20)] and hence the computational cost will remain linear. Furthermore, this approximation is well justified for the following two reasons:

1) Most of the elements of the vectors $\mathbf{k}_i$ have very small values, except (i) the elements that correspond to the robot, and (ii) elements corresponding to the landmarks strongly correlated with other landmarks that were observed over the last $(M_{max}-M_{min})$ time steps.
2) The rank-2 covariance update process (described in Section III-D), sparsifies the $\mathbf{k}_i$'s by replacing the largest elements, in the absolute value sense, of the $\mathbf{k}_i$'s with zeros. Hence, only few directions, $\mathbf{v}_i$, of $\mathbf{D}$ contain substantial information (typically $M_{min}=1$ or 2). The remaining ones can be discarded without significant loss of accuracy.

At this point, we should note that the SWKF approach in [Julier, 2001] is also based on the first observation mentioned above. In that case, however, all elements of $\mathbf{k}_i$, except those corresponding to the robot and the observed landmark, are discarded at *every* time step. Since there exist strong correlations between neighborhoods of landmarks in dense maps, this crude approximation generates very conservative updates in the SWKF. Furthermore, and in stark contrast to the one-step approximations involved

[8] At this point, we should remind the reader that the matrix $\mathbf{D}$ is never explicitly calculated. Instead the vectors $\mathbf{k}_i$ are stored for processing in the ensuing approximations.

in [Julier, 2001] and [Guivant and Nebot, 2001], by employing the GMP framework, we delay the time when an approximation becomes necessary. This, in effect, allows us to retain the most *informative* among all the $\mathbf{k}_i$ vectors accumulated over an extended period of time, thus significantly reducing the information loss.

A simplistic and very fast solution to the low-rank approximation described in (29) would be to select and retain the largest M_{min} out of the M_{max} available $\mathbf{k}_i$ vectors based on their 2-norm, $||\mathbf{k}_i||$. Although this is often a reasonable approximation and, as explained later, guarantees that the resulting estimator remains conservative, it is not optimal unless:

1) $\mathbf{k}_i = \sqrt{\lambda_i}\mathbf{v}_i$ for $i = 1 \dots M_{min}$, where $\mathbf{v}_i$ are the eigenvectors corresponding to the M_{min} largest eigenvalues of $\mathbf{D}$, or
2) $||\mathbf{k}_i|| \simeq 0$, for $i = (M_{min}+1) \dots M_{max}$.

While condition (1) is rarely satisfied in practice, condition (2), from extensive simulation studies, is seen to be usually true for $i = (M_{mid}+1) \dots M_{max}$, where $M_{mid} \gg M_{min}$.

Since the objective of the Power-SLAM estimator is to minimize the information loss, i.e., *minimize* the trace of the covariance matrix $\mathbf{P}_{k+m|k+m}$ [cf. (20)], it is necessary to *maximize* the trace of the approximated Kalman vector outer-product sum $\mathbf{D}^*$ [cf. (28)]. The optimal solution to this maximization problem can only be obtained by determining the eigenvectors $\mathbf{v}_i$ that correspond to the M_{min} largest eigenvalues of $\mathbf{D}$ and constructing $\mathbf{D}^*$ [cf. (29)]. Furthermore, in order to maintain the linear-time nature of our proposed approach, it is necessary to use an algorithm that calculates these eigenvalue-eigenvectors pairs in linear time. We next show how this can be accomplished by employing the Power method [Golub and Loan, 1996] (Algorithm 1).

Algorithm 1

Require: Matrix $\mathbf{D}$, scalars n_p, M_{min}
1: **for** $j = 1$ to M_{min} **do**
2: Generate random vector[9] $\mathbf{s}_0$
3: **for** $k = 0$ to $(n_p - 1)$ **do**
4: Compute $\mathbf{s}_{k+1} \leftarrow \mathbf{D}\mathbf{s}_k$
5: Find $\alpha \leftarrow ||\mathbf{s}_{k+1}||_\infty$
6: $\mathbf{s}_{k+1} \leftarrow \mathbf{s}_{k+1}/\alpha$
7: **end for**
8: $\lambda_j \leftarrow \alpha$ $\{\lambda_j$ is the dominant eigenvalue of $\mathbf{D}\}$
9: $\mathbf{v}_j \leftarrow \mathbf{s}_{n_p}/||\mathbf{s}_{n_p}||$ $\{\mathbf{v}_j$ is the eigenvector of $\mathbf{D}$, corresponding to $\lambda_j\}$
10: $\mathbf{D} \leftarrow \mathbf{D} - \lambda_j \mathbf{v}_j \mathbf{v}_j^T$
11: **end for**
12: **return** $\lambda_j, \mathbf{v}_j, j = 1, \dots, M_{min}$

In order to evaluate the computational complexity of the Power method, when applied to this problem, consider the first iteration when $j = 1$ and $k = 0$. Given $\mathbf{s}_0$, Step 4 of Algorithm 1 calculates:

$$\mathbf{s}_1 = \mathbf{D}\mathbf{s}_0 = \left(\sum_{i=1}^{M_{max}} \mathbf{k}_i {\mathbf{k}_i}^T \right) \mathbf{s}_0 = \sum_{i=1}^{M_{max}} \mathbf{k}_i (\mathbf{k}_i^T \mathbf{s}_0) \quad (30)$$

and the computational cost for this step is $O(M_{max}N)$. Next, the costs for obtaining the ∞-norm in Step 5 and dividing $\mathbf{s}_{k+1}$ by α in Step 6 are $O(N)$ each. Thus the total cost for Steps 4-6 remains $O(M_{max}N)$. Furthermore, since Steps 4-6 are repeated n_p times[10], the total cost to acquire α (dominant eigenvalue λ_1) and $\mathbf{s}_{n_p}$ (dominant eigenvector $\mathbf{v}_1$), becomes $O(n_p M_{max} N)$. Once the dominant eigenvalue/eigenvector is obtained, $\mathbf{D}$ is modified in Step 10 to include the additional vector outer-product, $\lambda_j \mathbf{v}_j \mathbf{v}_j^T$. As a result, $\mathbf{D}$ will now contain $M_{max}+1$ vector outer-products.

Similarly, by repeating the above process, the second largest eigenvalue/eigenvector pair can be acquired at a cost of $O(n_p(M_{max}+1)N)$ and the new $\mathbf{D}$ will contain $M_{max}+2$ vector outer-products. Thus, we can see that the cost of obtaining the i^{th} largest eigenvalue/eigenvector pair, where $i = 1, \dots, M_{min}$, is $O(n_p(M_{max} + (i-1))N)$. The total cost for obtaining all M_{min} such pairs becomes:

$$\begin{aligned} &O(n_p(M_{min}M_{max} + (1 + \dots + (M_{min}-1)))N) \\ &\quad = O(n_p M_{min}(M_{max} + \frac{M_{min}-1}{2})N) \\ &\quad \approx O(n_p M_{min} M_{max} N) \end{aligned}$$

since $M_{min} \ll M_{max}$. Hence, as long as $n_p M_{min} M_{max} \ll N$, the computational cost of the Power method remains linear in N.

Remark 1 (Speeding up the Power Method): In order to expand the time horizon over which this low-rank approximation is delayed (i.e., intuitively large values of M_{max} allow us to retain the most informative $\mathbf{k}_i$ vectors), a further approximation can be employed based on the condition (2) mentioned earlier, i.e., when the $\mathbf{k}_i$ are sorted by their 2-norm, it has been observed that $||\mathbf{k}_i|| \simeq 0$, for $i = (M_{mid}+1) \dots M_{max}$, where $M_{min} \ll M_{mid} \ll M_{max}$. Based on this observation, the matrix $\mathbf{D}$ is first approximated as:

$$\mathbf{D} = \sum_{i=1}^{M_{max}} \mathbf{k}_i \mathbf{k}_i^T \simeq \sum_{i=1}^{M_{mid}} \mathbf{k}_i \mathbf{k}_i^T = \tilde{\mathbf{D}} \quad (31)$$

and Algorithm 1 is applied to $\tilde{\mathbf{D}}$ instead of $\mathbf{D}$ to determine its M_{min} largest eigenvectors and eigenvalues, i.e.,

$$\tilde{\mathbf{D}} = \sum_{i=1}^{M_{mid}} \tilde{\lambda}_i \tilde{\mathbf{v}}_i \tilde{\mathbf{v}}_i^T \simeq \sum_{i=1}^{M_{min}} \tilde{\lambda}_i \tilde{\mathbf{v}}_i \tilde{\mathbf{v}}_i^T = \sum_{i=1}^{M_{min}} \tilde{\mathbf{k}}_i^* \tilde{\mathbf{k}}_i^{*T} = \tilde{\mathbf{D}}^*, \quad (32)$$

where $\tilde{\lambda}_i$, $\tilde{\mathbf{v}}_i$, and $\tilde{\mathbf{k}}_i^*$ are defined as in (29).

Selecting the M_{mid} largest (in the 2-norm sense) $\mathbf{k}_i$ vectors incurs a cost of $O(M_{max}N)$ for determining their magnitude and a cost of $O(M_{max}\log(M_{max}))$ for sorting the vectors in descending order based on their 2-norms. After this process is complete, the cost of the Power Method reduces to

[9] In this particular problem the convergence speed of the Power Method increases significantly by selecting $\mathbf{s}_0 = \mathbf{k}_j$ where $||\mathbf{k}_j|| > ||\mathbf{k}_i||$, $\forall i \in \{1 \dots M_{max}\} \backslash \{j\}$.

[10] In most cases, n_p = 7-10 steps are necessary for this iterative process to converge. Based on two successive estimates for the eigenvector, convergence is detected when $|1 - \mathbf{s}_{n_p}^T \mathbf{s}_{n_p-1}/(||\mathbf{s}_{n_p}|| \times ||\mathbf{s}_{n_p-1}||)| < 10^{-6}$, i.e, the angle between these two vectors is smaller than $\sim 10^{-6}$ rad.

$O(n_p M_{min} M_{mid} N)$. Typical values of these parameters used in our tests are: (i) $M_{max} = (2 - to - 10)\%$ of N, (ii) $M_{mid} = \max\{2, (5 - to - 10)\%$ of $M_{max}\}$ (i.e., the Power Method is not used when $M_{mid} = 2$ which corresponds to $N < 250$, i.e, ~ 100 -landmark maps), (iii) $M_{min} = 1 - 2$, and (iv) $n_p = 7 - 10$ (i.e., $n_p M_{min} M_{mid}$ is 1-2 orders of magnitude smaller than N). Note that these are only representative values and they can be adjusted on-line to meet the availability of computational resources.

Remark 2 (Conservative Estimator): A key advantage of the presented low-rank approximation is that the covariance matrix remains conservative. Since $\mathbf{D} \succeq \mathbf{D}^*$ [cf. (29)], (20) yields:

$$\mathbf{P}_{k+m|k+m} = \mathbf{P}_{k+m|k} - \mathbf{D} \;\succeq\; \mathbf{P}_{k+m|k} - \mathbf{D}^* = \mathbf{P}^*_{k+m|k+m} \tag{33}$$

Here $\mathbf{P}^*_{k+m|k+m} = \mathbf{P}_{k+m|k} - \sum_{i=1}^{M_{min}} \mathbf{k}^*_i \mathbf{k}^{*T}_i$ is the new covariance. The estimator also remains conservative for $\tilde{\mathbf{D}}^*$, since [cf. (31), (32)] $\mathbf{D} \succeq \tilde{\mathbf{D}} \succeq \tilde{\mathbf{D}}^*$.

Remark 3 (Quantifying the Information Loss): Importantly, the Power-SLAM approach provides a concrete measure of the information loss incurred due to the low-rank approximation. Quantifying the approximation involved is necessary in order to adjust the parameters M_{min} and M_{max} on-line, so as to meet performance requirements. This can be achieved by computing the ratio $(\mathrm{tr}(\mathbf{D}) - \mathrm{tr}(\mathbf{D}^*))/\mathrm{tr}(\mathbf{D})$ with complexity $O((M_{max} + M_{min})N)$, where $\mathrm{tr}(\mathbf{D}) = \sum_{i=1}^{M_{max}} \mathbf{k}_i{}^T \mathbf{k}_i$ and $\mathrm{tr}(\mathbf{D}^*) = \sum_{i=1}^{M_{min}} \mathbf{k}^{*T}_i \mathbf{k}^*_i$.

D. Linear-Time, Rank-2 Covariance Updates

The main drawback of any low-rank approximation of $\mathbf{D}$ [cf. (29)] is that it does not guarantee loss of rank of the covariance matrix, $\mathbf{P}_{k+m|k+m}$ [cf. (33)], after "infinite" time, as is expected when the system reaches steady-state [Dissanayake et al., 2001], [Mourikis and Roumeliotis, 2006]. This is due to the fact that the rank of matrix $\mathbf{P}_{k+m|k}$, in general, is $(2N + 3)$, i.e., it is full-rank, while the rank of $\mathbf{D}$ is at most M_{max}. Hence the rank of $\mathbf{P}_{k+m|k+m} = \mathbf{P}_{k+m|k} - \mathbf{D}$ will be at least $(2N + 3 - M_{max})$ with $M_{max} \ll N$. Furthermore, due to the propagation steps [cf. (21)], the covariance $\mathbf{P}_{k+m|k}$, in general, will increase continuously. The same is true for $\mathbf{D} = \sum_{i=1}^{2m} \mathbf{k}_i \mathbf{k}_i{}^T$, where $\mathbf{D}$ will also become larger when the same landmarks are re-observed in a given period of time. To overcome this drawback, we need to guarantee that the trace of $\mathbf{P}_{k+m|k}$ will decrease monotonically. This can be achieved by subtracting certain elements of $\mathbf{D}$ from $\mathbf{P}_{k+m|k}$ at every time step. However, note that any modification of $\mathbf{D}$ requires that the positive semi-definite property of $\mathbf{D}$ be maintained, else the low-rank approximation described in the previous section will not guarantee consistency. In order to achieve this, we propose the following rank-2 covariance updates:

$$\begin{aligned}\mathbf{P}_{k+m|k+m} &= \mathbf{P}_{k+m|k} - \sum_{i=1}^{2m} \mathbf{k}_i \mathbf{k}_i{}^T \\ &= \left(\mathbf{P}_{k+m|k} - \delta\mathbf{P}_j\right) - \left(\mathbf{k}^+_j \mathbf{k}^{+T}_j + \sum_{i=1, i\neq j}^{2m} \mathbf{k}_i \mathbf{k}_i{}^T\right) \\ &= \mathbf{P}_{k+m|k+1} - \sum_{i=1}^{2m} \mathbf{k}^+_i \mathbf{k}^{+T}_i \end{aligned} \tag{34}$$

where $\mathbf{k}^+_i = \mathbf{k}_i$, $\forall i \neq j$, $\mathbf{k}^+_j = (\mathbf{I} - \mathbf{A}_j)\mathbf{k}_j$,

$$\delta\mathbf{P}_j = (\mathbf{A}_j\mathbf{k}_j)(\mathbf{A}_j\mathbf{k}_j)^T + \mathbf{k}^+_j(\mathbf{A}_j\mathbf{k}_j)^T + (\mathbf{A}_j\mathbf{k}_j)\mathbf{k}^{+T}_j \tag{35}$$

and $\mathbf{k}_j\mathbf{k}_j{}^T = \delta\mathbf{P}_j + \mathbf{k}^+_j\mathbf{k}^{+T}_j$. In the above expressions, $\mathbf{A}_j$ is a selector matrix, $\mathbf{k}_j$ is the vector used in the update (to be determined), and $\mathbf{P}_{k+m|k+1}$ denotes the updated covariance matrix after incorporating a single rank-2 covariance update. Furthermore, as required, after the rank-2 covariance update, the new matrix $\mathbf{D}^* = \sum_{i=1}^{2m} \mathbf{k}^+_i \mathbf{k}^{+T}_i$ remains positive semi-definite, since it is still expressed as the accumulated sum of vector outer-products. In order to carry out this rank-2 covariance update, $\mathbf{k}_j$ and $\mathbf{A}_j$ need to be determined such that the following two constraints are satisfied:

(C1) The cost of computing $\delta\mathbf{P}_j$ [cf. (35)] is minimized, allowing at most $O(N)$ operations to maintain the linear computational complexity of the algorithm.

(C2) The trace of $\mathbf{P}_{k+m|k+1} = \mathbf{P}_{k+m|k} - \delta\mathbf{P}_j$ is minimized. Note that the minimization of $\mathrm{tr}(\mathbf{P}_{k+m|k+1})$ ensures minimization of $\mathrm{tr}(\mathbf{P}_{k+m|k+m})$ when the vectors $\mathbf{k}_i$ are discarded during the low-rank approximation.

Since the vector $\mathbf{k}_j$ can, in general be dense, while computing $\mathbf{A}_j\mathbf{k}_j$, (C1) requires that the matrix $\mathbf{A}_j$ has at most $n \ll N^2$ non-zero elements[11]. If these non-zero elements are distributed among $1 \leq p \leq N$ rows of $\mathbf{A}_j$, then the cost for computing $\mathbf{A}_j\mathbf{k}_j$ is $O(n)$ and the resulting vector will have p non-zero elements. Since $(\mathbf{A}_j\mathbf{k}_j)(\mathbf{A}_j\mathbf{k}_j)^T$ is a symmetric matrix, the cost for computing it will be $\frac{p(p+1)}{2}$.

When computing $\mathbf{k}^+_j = (\mathbf{I} - \mathbf{A}_j)\mathbf{k}_j = \mathbf{k}_j - \mathbf{A}_j\mathbf{k}_j$, d, $d \in \{0, 1, \ldots, p\}$ subtractions are necessary, depending on the number of elements of $\mathbf{k}^+_j$ that can be directly set to zero (i.e., by appropriately selecting the elements of $p - d$ rows of $\mathbf{A}_j$, it can ensured that elements of $\mathbf{A}_j\mathbf{k}_j$ in these rows are same as those of $\mathbf{k}_j$). Thus, if $\mathbf{k}^+_j$ contains $p - d$ zeros, computing $\mathbf{k}^+_j(\mathbf{A}_j\mathbf{k}_j)^T$ requires $(N - (p - d))p$ operations. Hence the total cost for calculating $\delta\mathbf{P}_j$ can be expressed as a function of p, d, n and N as follows:

$$c(p, d, n, N) = \frac{1}{2}\left(-p^2 + (2N + 2d + 1)p + 2(n + d)\right). \tag{36}$$

Note that (36) is a concave function of p with the maximum achieved at $p = \frac{2N+2d+1}{2} > N$. Thus, it is a monotonically increasing function within the interval of interest, i.e., $[1 \ldots N]$, with the minimum occurring at $p = 1$ (i.e., since $\mathbf{A}_j$ cannot be a matrix of all zeros, at least one row of $\mathbf{A}_j$ will have

[11] For clarity in the following derivations, we set the state vector size to N.

non-zero elements). Substituting $p = 1$ in (36) the total cost becomes:

$$c(1, d, n, N) = N + n + 2d. \tag{37}$$

Now since $p = 1$, the number of subtraction d can either be 0 or 1. Also, the structure of matrix $\mathbf{A}_j$, that contains only one non-zero row (e.g., the ξ^{th} row), is given by:

$$\mathbf{A}_j^T = [\mathbf{0} \dots \mathbf{a}_\xi \dots \mathbf{0}], \tag{38}$$

where $\mathbf{a}_\xi$ denotes the ξ^{th} row of $\mathbf{A}_j$, with $n \leq N$ non-zero elements.

We now turn our attention to (C2). Minimizing the trace of $\mathbf{P}_{k+m|k+1}$ is equivalent to maximizing the trace of $\delta\mathbf{P}_j$. Substituting (38) in (35), we obtain:

$$\begin{aligned} \mathrm{tr}(\delta\mathbf{P}_j) &= \mathbf{k}_j^T \left(\mathbf{A}_j + \mathbf{A}_j^T - \mathbf{A}_j^T\mathbf{A}_j\right)\mathbf{k}_j \\ &= -(\mathbf{a}_\xi^T\mathbf{k}_j)^2 + 2k_{\xi j}(\mathbf{a}_\xi^T\mathbf{k}_j) \end{aligned} \tag{39}$$

where $k_{\xi j}$ is the ξ^{th} scalar element of vector $\mathbf{k}_j$ and (39) is a concave function of $\mathbf{a}_\xi$. Computing its derivative with respect to the elements of $\mathbf{a}_\xi$, the maximum of $\mathrm{tr}(\delta\mathbf{P}_j)$ is reached when:

$$(\mathbf{a}_\xi^T\mathbf{k}_j)\mathbf{k}_j = k_{\xi j}\mathbf{k}_j. \tag{40}$$

This is trivially achieved by setting $\mathbf{a}_\xi = \mathbf{e}_\xi$, where $\mathbf{e}_\xi$ is the ξ^{th} canonical unit vector. Therefore, $\mathbf{A}_j$ [cf. (38)] becomes a matrix of zeros, except the ξ^{th} diagonal element which is equal to one. As a result of this, the vector $\mathbf{A}_j\mathbf{k}_j$ has only one non-zero element, i.e., $k_{\xi j}$, in the ξ^{th} location; the rest of its elements are zero. Also, the vector $\mathbf{k}_j^+$ has the same elements as $\mathbf{k}_j$, except the ξ^{th} element, which is zero.

Finally from (35), we can see that $\delta\mathbf{P}_j$ will have non-zero elements only in its ξ^{th} row and column. Hence, the total cost for computing $\delta\mathbf{P}_j$ becomes $c(1, 0, 0, N) = N$. Subtracting $\delta\mathbf{P}_j$ from $\mathbf{P}_{k+m|k+m}$ will also have cost N. Moreover, due to this special structure of the resulting $\delta\mathbf{P}_j$ matrix (i.e., non-zero elements only in its ξ^{th} row and column), the rank of this matrix is 2 (hence the name rank-2 updates).

What remains to be determined are the indices j and ξ that satisfy (C2). Substituting $\mathbf{a}_\xi^T\mathbf{k}_j = k_{\xi j}$ in (39), we have $\max(\mathrm{tr}(\delta\mathbf{P}_j)) = k_{\xi j}^2$. Hence maximizing the $\mathrm{tr}(\delta\mathbf{P}_j)$ is guaranteed by selecting among the $\mathbf{k}_j$ vectors, the one which has the maximum element $k_{\xi j}$, in the absolute value sense. This maximum element, among $2m$ Kalman vectors (each of dimension $N \times 1$), can be determined at a cost of $O(mN)$.

Thus, we demonstrated that the overall computational complexity of a single rank-2 covariance update is $O(mN)$. Furthermore, this rank-2 covariance update process can be repeated multiple times during each time step, depending on the availability of computational resources, to further decrease the trace of $\mathbf{P}_{k+m|k+1}$ and speed up convergence.

Before presenting the simulation and experimental results, we summarize the three key algorithmic components of our proposed approach along with their computational complexity [cf. Table III]. Firstly, we showed that by using the GMP technique, approximations necessary for ensuring linear computational complexity of EKF-based SLAM can be delayed over multiple time steps. Secondly, we presented a linear-cost low-rank approximation technique that retains the most informative Kalman vectors from the postponement phase using the Power Method. Lastly, in order to speed up the convergence of our proposed estimator, linear-complexity rank-2 covariance updates were introduced. Depending on the availability of computational resources at each time step, multiple rank-2 updates can be carried out to further speed up convergence.

TABLE III
COMPUTATIONAL COMPLEXITY OF POWER-SLAM. N: NUMBER OF LANDMARKS IN THE MAP, m: NUMBER OF KALMAN VECTORS IN THE VECTOR OUTER-PRODUCT SUM, $M_{min} \leq m \leq M_{max} \ll N$, n_p: NUMBER OF ITERATIONS OF THE POWER METHOD

Steps in Power-SLAM	Computational Complexity
Global Map Postponement	$O(mN)$
Low Rank Approximation (Power Method)	$O(n_p M_{min} M_{max} N)$
Linear-time Rank-2 updates	$O(mN)$

IV. SIMULATIONS

A. Simulation Setup

The simulations used to validate the performance of the Power-SLAM algorithm have been implemented in MATLAB. The robot starts at a known position and follows an 8-shaped trajectory shown in Fig. 1, where the radius of each circle is 150 m. The maximum sensing range of the robot is set to 8 m and it has a 360 degrees field of view for range and bearing measurements. The noise in the measurements is modeled as zero-mean, white Gaussian. Every 0.2 seconds, the robot receives the following measurements: (i) odometry (linear, v, and rotational, ω, velocity) with noise standard deviation $\sigma_v = 3\%v$, and $\sigma_\omega = 3\%\omega$, (ii) range d, with $\sigma_d = 8$ cm, and (iii) bearing θ, with $\sigma_\theta = 1$ degree.

In this simulation, the robot observes approximately 500 landmarks (i.e., the size of the state vector increases from 3 to 1000) over 2000 time steps with an average of 1.6 landmark observations per time step. The robot closes loops approximately every 310 time steps. The maximum number of Kalman vectors, M_{max}, and the number of rank-2 updates at each time step, are both set to 10% of the size of the state vector at that time step. The number of Kalman vectors, $\mathbf{k}_i$, considered for the low-rank approximation, are set to $M_{mid} = \max(2, 0.05M_{max})$. The Power Method extracts the dominant eigenvalue and eigenvector ($M_{min} = 1$).

B. Simulation Results

The objective of our simulation studies is to demonstrate the accuracy of the Power-SLAM algorithm, verify its consistency, and compare its performance to that of (i) EKF-based SLAM, (ii) SWKF SLAM [Julier, 2001], and (iii) CKF SLAM [Guivant and Nebot, 2001]. Note that the standard EKF-SLAM has computational complexity $O(N^2)$, while all other algorithms evaluated hereafter have processing requirements linear, $O(N)$, in the number of features. However, there are certain differences in the actual processing cost of each of the linear estimators. Although the SWKF estimator has fixed processing requirements, the CKF computational cost can be adjusted. To ensure a fair comparison, the CKF covariance

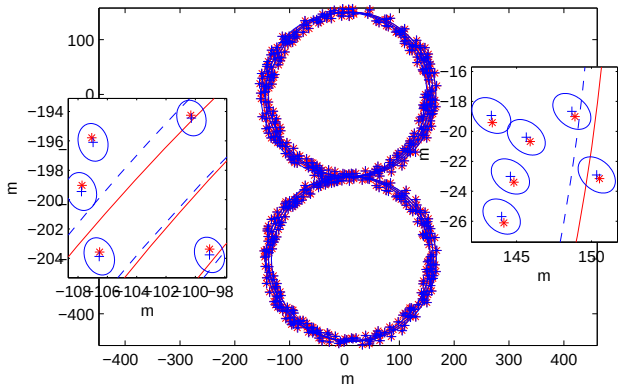

Fig. 1. True robot trajectory (solid red line), true landmark positions (*), Power-SLAM estimated robot trajectory (dashed blue line), Power-SLAM estimated landmark positions (+), and their 3σ uncertainty ellipses. Insets are zoomed sections for better viewing of the uncertainty ellipses.

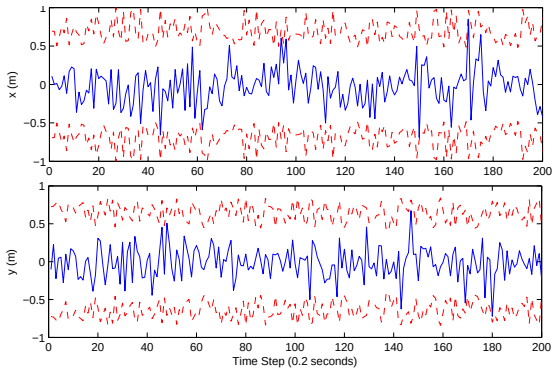

Fig. 2. Measurement residuals (solid) and corresponding 3σ bounds (dashed).

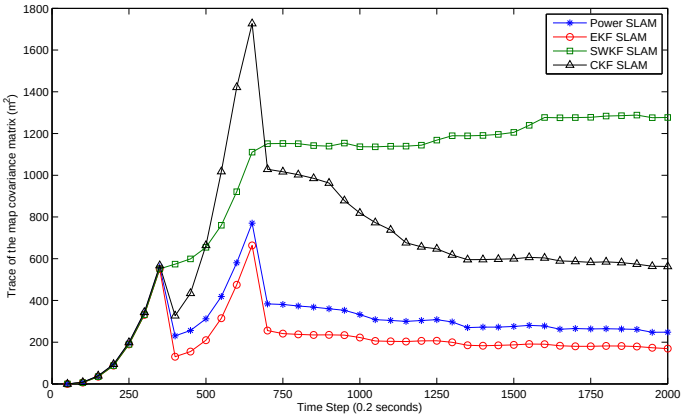

Fig. 4. Trace of the map's covariance matrix.

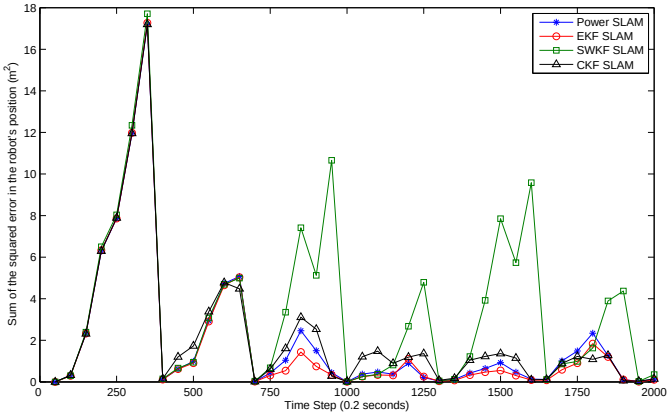

Fig. 5. Sum of the squared error in the robot's position estimates.

updates are set so as to have the same cost as the rank-2 updates of the Power-SLAM algorithm.

We start with a qualitative evaluation of the Power-SLAM algorithm. As shown in Fig. 1, the Power-SLAM estimates for both the robot trajectory and the landmark positions are very close to the real ones. Also note that the 3σ ellipses of uncertainty for the estimated landmark positions contain the true positions, indicating consistency. Fig. 2 depicts the measurement residuals along with their corresponding 3σ bounds for the Power-SLAM method (only 200 time steps are shown to ensure clarity). This figure verifies that the Power-SLAM estimator is consistent.

We now turn our attention to the quantitative results presented in Figs. 3-6. Although all 3 linear-complexity estimators are conservative as compared to the standard EKF-SLAM, the SWKF is the most conservative one, followed by the CKF. The Power-SLAM estimator is the least conservative, which is evident when comparing the trace of the robot-position covariance matrix to the corresponding one for the EKF-

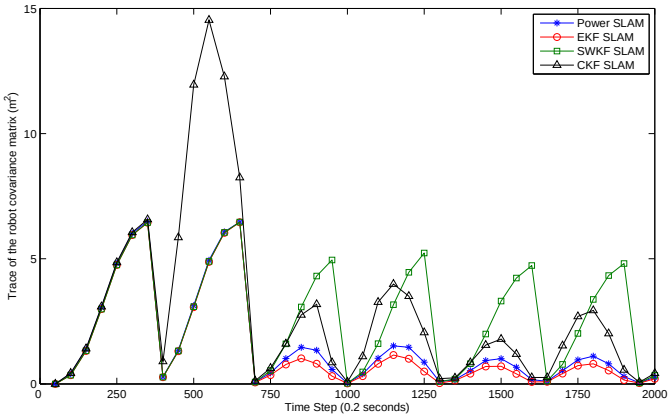

Fig. 3. Trace of the robot's covariance matrix.

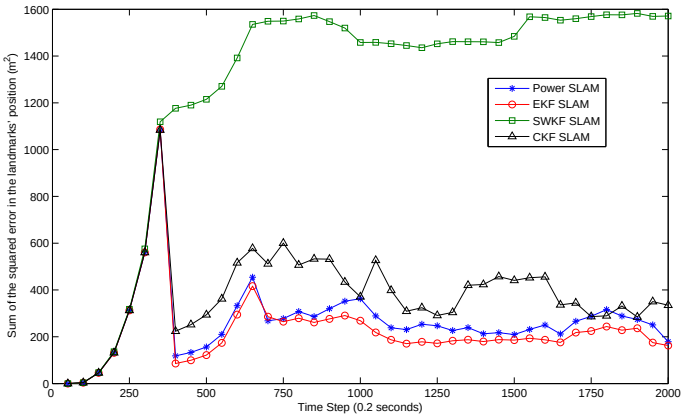

Fig. 6. Sum of the squared error in the landmarks' position estimates.

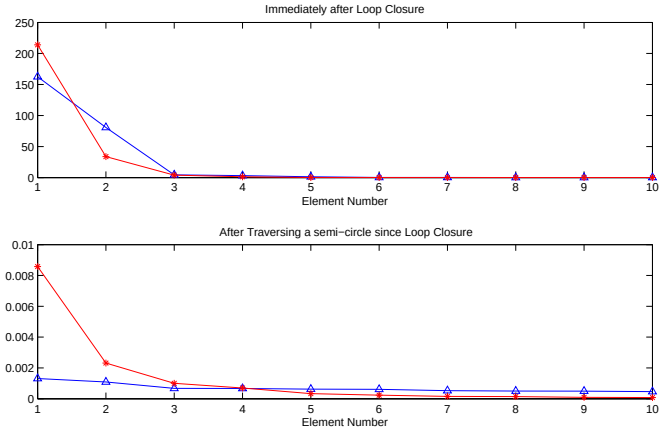

Fig. 7. The 10 largest squared 2-norms (△) of the Kalman vectors, and the 10 largest eigenvalues (*) of the Kalman vector outer-product sum at 2 time instances: immediately after loop closure (top figure) and traversing a semi-circle after loop closure (bottom figure).

SLAM [cf. Fig. 3]. The same conclusion can be reached by comparing the traces of the landmarks' covariance matrices for each of these estimators [cf. Fig. 4]. For the case of the landmarks, in particular, the SWKF covariance does not decrease with time as the robot revisits the same areas. While this is not true for the CKF, the rate of decrease of the covariance matrix trace is very slow when compared to that of the Power-SLAM estimator. This behavior is due to the fact that both the SWKF and the CKF are based on crude approximations that take place during each time step and result in large information loss. In contrast, the Power-SLAM algorithm is able to minimize the information loss by (i) delaying approximations over large time horizons, and (ii) extracting and retaining the most informative Kalman vectors during each approximation.

The level of "conservatism" of each algorithm, when compared to the EKF-SLAM estimator, also affects the accuracy of the estimates. Specifically, both the robot's and landmarks' position errors for the SWKF and CKF are significantly larger when compared to the ones for the Power-SLAM algorithm (Figs. 5 and 6), which achieves accuracy almost indistinguishable to that of EKF-SLAM.

The average squared error in the position estimates for each landmark, when compared to the standard EKF, is 72% higher for the CKF and 483% for the SWKF, whereas it is only 16% higher for Power-SLAM. Similarly, for the robot position estimates, the average squared error, when compared to the standard EKF, is 17% higher for the CKF and 94% for the SWKF while it is only 5% higher for Power-SLAM. This is due to the fact that the Power-SLAM algorithm is based on optimal approximations within the linear-complexity processing constraints.

Fig. 7 shows the 10 largest eigenvalues and 10 largest values (in the squared 2-norm sense) of the 100 Kalman vectors in $\mathbf{D}$ at two time instances: (i) just after loop closure, and (ii) when the robot has traveled a semi-circle after loop closure. As expected, the Kalman vectors carry substantially more information after loop closure than at other time steps. Moreover, in both cases, 2% to 10% of the Kalman vectors carry the bulk of the information and hence the others can be discarded in order to speed up the Power Method as discussed in Remark 1.

V. Experiments

A. Experimental Setup

An iRobot Packbot robot, equipped with a Pointgrey Firefly stereo rig and a PC104 computer was used for the experiments [cf. Fig. 8]. The stereo rig has been calibrated using the calibration technique by Zhang [Zhang, 2000] and Heikkila *et al.* [Heikkila and O.Silven, 1997] to obtain its intrinsic and extrinsic parameters.

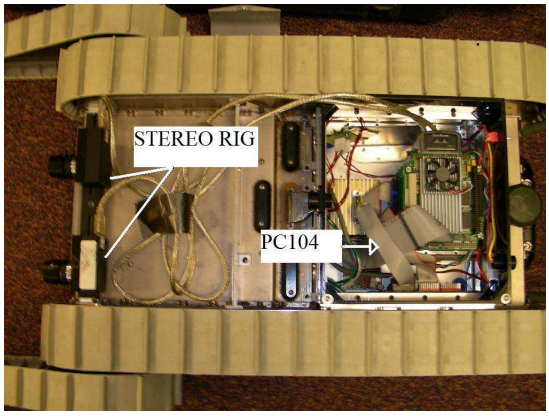

Fig. 8. The Packbot robot equipped with a Pointgrey Firefly stereo rig.

During the experiments, the Packbot explored an indoor office environment and captured stereo images of its surroundings while moving in a plane. The robot received proprioceptive measurements, i.e., linear, v_m, and angular, ω_m, velocity, at 10 Hz and exteroceptive measurements, i.e., the stereo images, at approximately $0.5 - 1$ Hz. The noise in the proprioceptive measurements is assumed to be zero-mean, white Gaussian with standard deviation $\sigma_v = 3\%\ max(v_m)$ and $\sigma_\omega = 3\%\ max(\omega_m)$. The image resolution is 640×480 and an additive white Gaussian noise of 2 pixels is assumed for the camera measurements.

For the duration of the experiment (approximately 2.5 mins), a total of 103 images were captured by each camera. SIFT keypoints [Lowe, 2004], matched in the corresponding stereo images, are used to determine the 3D position of the point features based on stereo triangulation. Examples of detected and matched keypoints are shown in Figs. 9 and 10, respectively. On average, 1247 keypoints/image were detected, while 14.36 keypoints were matched for each set of stereo images. A total of 1088 point features were added to the state vector.

B. Experimental Results

Fig. 11 compares the trace of the robot's position covariance matrix for the SWKF, CKF, EKF-SLAM and the Power-SLAM estimators. Note that the trace of the robot's covariance matrix for the Power-SLAM estimator is closest to that of the EKF as compared to the SWKF and CKF. Thus, we

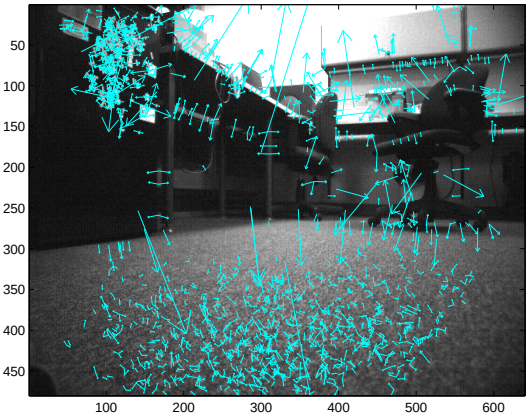

Fig. 9. Example image with detected SIFT keypoints.

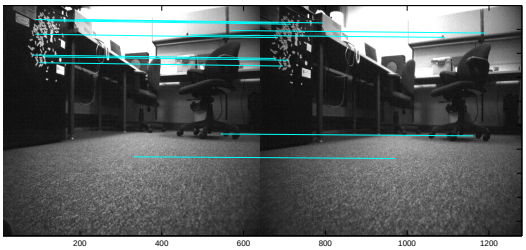

Fig. 10. Example of matched SIFT keypoints in the left and right images.

can conclude that the Power-SLAM estimator is the least conservative approximate estimator, followed by the CKF and finally the SWKF. Fig. 12 compares the trace of the features' covariance matrix for the aforementioned four estimators. Here, only the covariance for features that have been re-observed has been included for comparison. From this figure, we see that the performance of the Power-SLAM estimator is almost indistinguishable from that of the EKF. Furthermore, with respect to the uncertainty in the features' position estimates, the Power-SLAM estimator is the least conservative as compared to the SWKF and the CKF. Thus, we conclude that by employing the Global-Map Postponement technique and the Power Method, the Power-SLAM estimator minimizes the information loss while satisfying the linear-complexity constraint, and outperforms competing linear-processing cost alternatives.

VI. Conclusions

The Power-SLAM algorithm, introduced in this paper, provides a real-time consistent estimator for simultaneous localization and mapping that has computational complexity linear in the number of features in the map. The Global-Map Postponement approach followed by the Power Method and linear-time rank-2 updates form the crux of the Power-SLAM algorithm. The Global-Map Postponement technique delays the approximations over multiple time steps. The Power Method extracts and retains the dominant information from the Kalman vectors generated during the postponement phase.

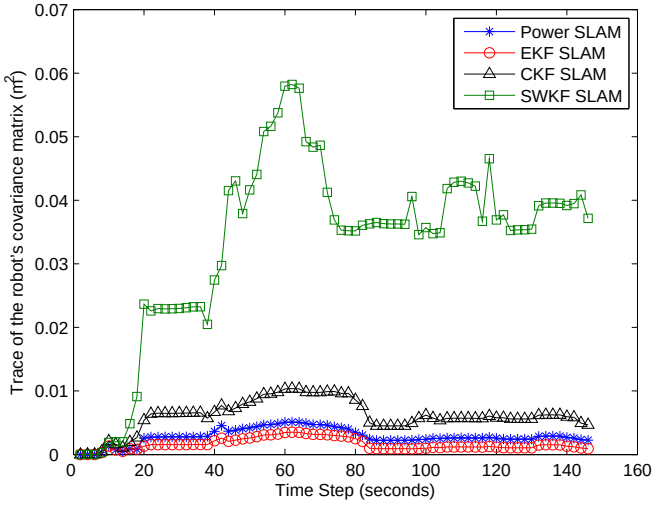

Fig. 11. Trace of the robot's position covariance matrix.

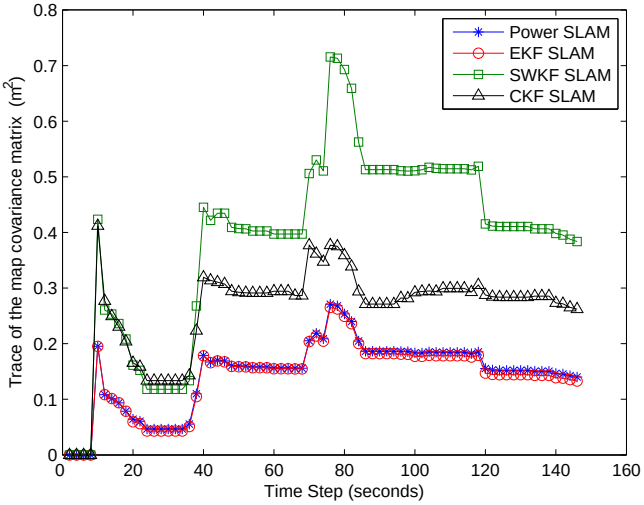

Fig. 12. Trace of the map's covariance matrix.

By working in tandem, these two techniques *minimize* the information loss over *multiple* time steps. Finally, in order to increase the convergence rate of this estimator, linear-time rank-2 updates, which minimize the trace of the covariance matrix, are applied at every time step. One of the key advantages of the Power-SLAM estimator is its ability to adjust its processing requirements on-line to meet the availability of computational resources. By adaptively trading CPU cycles for estimation accuracy, Power-SLAM bridges the gap between linear-complexity estimators (based on coarse approximations, such as the SWKF and the CKF) and the quadratic-complexity optimal EKF-based SLAM. Furthermore, by minimizing the information loss induced during the necessary approximations, the Power-SLAM algorithm is able to maximize estimation accuracy for the exact same number of operations. The simulation and experimental results have shown that both the robot and map estimates computed by the Power-SLAM estimator

closely follow those of the standard EKF-SLAM and clearly outperform, in terms of accuracy, both the SWKF and the CKF.

ACKNOWLEDGMENT

This work was supported by the University of Minnesota (DTC), and the National Science Foundation (IIS-0835637, IIS-0643680, IIS-0811946).

REFERENCES

[Bar-Shalom et al., 2001] Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. (2001). *Estimation with applications to tracking and navigation*. New York: Wiley.

[Csorba and Durrant-Whyte, 1997] Csorba, M. and Durrant-Whyte, H. F. (1997). New approach to map building using relative position estimates. In *Proc. of SPIE*, volume 3087, pages 115–125, Orlando, FL.

[Davison, 1998] Davison, A. (1998). *Mobile Robot Navigation using Active Vision*. PhD thesis, Oxford University, Department of Engineering Science.

[Dellaert and Kaess, 2006] Dellaert, F. and Kaess, M. (2006). Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25(12):1181–1203.

[Dissanayake et al., 2000] Dissanayake, G., Durrant-Whyte, H., and Bailey, T. (2000). A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 1009–1014, San Francisco, CA.

[Dissanayake et al., 2001] Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., and Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241.

[Durrant-Whyte et al., 2000] Durrant-Whyte, H., Dissanayake, G., and Gibbens, P. (2000). Towards deployment of large-scale simultaneous localization and map building (SLAM) systems. Technical report, Australian Centre for Field Robotics, University of Sydney.

[Frese, 2006] Frese, U. (2006). Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122.

[Frese et al., 2005] Frese, U., Larsson, P., and Duckett, T. (2005). A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207.

[Golub and Loan, 1996] Golub, G. H. and Loan, C. F. V. (1996). *Matrix Computations*. Johns Hopkins University Press.

[Guivant and Nebot, 2001] Guivant, J. and Nebot, E. (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257.

[Heikkila and O.Silven, 1997] Heikkila, J. and O.Silven (1997). A four-step camera calibration procedure with implicit image correction. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, San Jaun, Puerto Rico.

[Julier, 2001] Julier, S. (2001). A sparse weight kalman filter approach to simultaneous localisation and map building. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1251–1256, Maui, HI.

[Julier and Uhlmann, 2001] Julier, S. and Uhlmann, J. (2001). Simultaneous localisation and map building using split covariance intersection. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1257–1262, Maui, HI.

[Knight et al., 2001] Knight, J., Davison, A., and Reid, I. (2001). Towards constant time SLAM using postponement. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 405–413, Maui, HI.

[Leonard and Feder, 2000] Leonard, J. and Feder, H. (2000). A computationally efficient method for large-scale concurrent mapping and localization. In *Proc. 9th International Symposium on Robotics Research*, pages 169–176. Springer-Verlag.

[Lowe, 2004] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110.

[Montemerlo et al., 2002] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: a factored solution to the simultaneous localization and mapping problem. In *18th National Conference on Artificial Intelligence*, pages 593–598, Menlo Park, CA.

[Mourikis and Roumeliotis, 2006] Mourikis, A. and Roumeliotis, S. (2006). Analytical characterization of the accuracy of SLAM without absolute orientation measurements. In *Proc. Robotics Science and Systems Conference*, pages 215–222, Philadelphia, PA.

[Moutarlier and Chatila, 1989] Moutarlier, P. and Chatila, R. (1989). Stochastic multi-sensory data fusion for mobile robot location and environmental modeling. In *5th International Symposium on Robotics Research*, pages 85–94, Tokyo.

[Nerurkar and Roumeliotis, 2007] Nerurkar, E. D. and Roumeliotis, S. I. (2007). Power-slam: A linear-complexity, consistent algorithm for slam. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 636–643, San Diego, CA.

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

[Paskin, 2003] Paskin, M. A. (2003). Thin junction tree filters for simultaneous localization and mapping. In *Proc. 18th International Joint Conference on Artificial Intelligence*, pages 1157–1164, Acapulco, Mexico.

[Smith et al., 1990] Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, New York, NY, USA.

[Uhlmann et al., 1997] Uhlmann, J. K., Julier, S. J., and Csorba, M. (1997). Nondivergent simultaneous map building and localization using covariance intersection. In *Proc. SPIE*, volume 3087, pages 2–11, Orlando, FL.

[Williams et al., 2002] Williams, S., Dissanayake, G., and Durrant-Whyte, H. (2002). An efficient approach to the simultaneous localisation and mapping problem. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 406–411, Washington, DC.

[Zhang, 2000] Zhang, Z. (2000). A flexible new technique for camera calibration. In *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 1330–1334.