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

Esha D. Nerurkar and Stergios I. Roumeliotis
Department of Computer Science and Engineering, University of Minnesota

***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 is based on three key ideas. Firstly, by introducing the Global-Map Postponement method, approximations necessary for ensuring linear computational complexity are delayed over many 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 in effect minimizes the information loss during each approximation epoch. Finally, linear-complexity, rank-2 updates, which minimize the trace of the covariance matrix, are applied to increase the speed of convergence of the estimator. In addition to being consistent, the resulting estimator has processing requirements that can be adjusted to the availability of computational resources. Simulation 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.**

I. Introduction

Mobile robot Simultaneous Localization and Mapping (SLAM) has been studied extensively in the literature and numerous solutions exist that differ, primarily, in the assumptions made for the environment, the map representation, the sensors, and the estimation framework considered, to name a few. In terms of the type of estimator used, many approaches employ the Extended Kalman Filter (EKF). The main reason for this is that, in addition to being the optimal[1] estimator, in the Minimum Mean Square Error (MMSE) sense, the EKF recursively computes a concrete measure of the *uncertainty* in the state estimates, namely the covariance matrix. This information is necessary for minimizing the risk of failure when making decisions (e.g., data association, path planning, etc). Unfortunately, storing and updating the covariance matrix is also the main limitation of EKF-based SLAM. Under the assumption that only few map features are detected at each time step, both the memory and computational complexity are quadratic $O(N^2)$ in the number of features, N, in the map. Even though the storage requirements can be handled efficiently by the devices available today, the computational complexity has prevented the deployment of mobile robots in large scale environments.

As detailed in the following section, a number of EKF-based approaches exists that address the computational complexity of SLAM by either delaying the quadratic covariance update step or employing an approximate structure for the estimator. The main limitation of the approaches under the first category, is that inevitably at some point, the covariance will need to be updated, which will incur $O(N^2)$ computational cost. For large values of N this can be impractical. On the other hand, many of the suboptimal approaches do not guarantee consistency of the approximate estimates, which can lead to divergence of the EKF. Furthermore, amongst the consistent suboptimal approaches, information is discarded during *every* time step, often based on criteria that do *not* guarantee the best use of the available CPU cycles.

To address this problem, in this paper we describe an EKF-based algorithm for SLAM that has computational complexity *linear* in the number of features in the map. The resulting approximate estimator, which is provably *consistent*, *minimizes the information discarded* over *multiple time steps*. This is achieved by: (i) Extending the time horizon over which approximations are invoked (Section III-B), and (ii) Computing and retaining, after each approximation, these Kalman vectors whose outer product minimizes the trace of the covariance matrix (Section III-C). Finally, rank-2 covariance updates are applied at every time step (Section III-D), which minimize the trace of the covariance matrix under the linear computational complexity constraint. An additional benefit of the presented method is 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 proposed algorithm, we hereafter briefly review some of the representative approaches to EKF-based SLAM.

II. Related Work

In their seminal work [20], Smith, Self and Cheeseman 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 [17] 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:[2]

[1] Up to linearization errors.

[2] Although numerous approaches exist for reducing the computational complexity of SLAM, e.g., Particle Filter [15], thin junction trees [19], treemaps and multigrids [7], [8], square root SAM [3], 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.

A. Optimal EKF-based SLAM

By restructuring the EKF equations, Davison [2] 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. Joss Knight *et al.* [13] extended this idea to the case of sub-maps. As in [2], covariance and state updates can be 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 [22], new sub-maps, initialized at various locations along the trajectory, are updated in constant time (i.e., quadratic in the number of features in this particular sub-map). However, once all the sub-maps need to be merged, the computational cost again becomes quadratic in the *total* number of features.

B. Approximate EKF-based SLAM

The method described in [4], [6] retains the optimal structure of the 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 is consistent and 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 [14] introduced the concept of multiple overlapping sub-map regions (Decoupled Stochastic Maps), each with its own stochastic map. Their approach scales the SLAM algorithm to linear time. 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 [1] 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 [18] 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 [10] 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 [2], [13], 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 [21] which does not consider the estimates' correlations. Although this estimator is consistent 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) [12] 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) [11] 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 is provably consistent and has linear computational complexity.

The approach presented in this paper reformulates and extends the postponement method [2], [13] to the case of the *global* map. We show that the computational cost in this case is *linear* in the number of states, N, for as long as the number of delayed updates, m, is significantly smaller than N. This is achieved by periodically employing a low-rank approximation of the sum of the Kalman vector outer products that needs to be subtracted from the covariance matrix. This process relies on the Power method [9] for computing the maximum eigenvalues/eigenvectors in linear time. Furthermore, it guarantees consistency, retains the most informative of these vectors, and allows us to extend the postponement horizon indefinitely. Finally, linear-cost, rank-2 updates, selected so as to minimize the trace of the covariance matrix, are imposed at every time step .

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. Vectors are denoted in bold type (e.g., $\mathbf{x}$) and subscripts indicate whether a quantity is related to the robot r or a landmark i, where $i = 1 \ldots N$ (subscripts will be omitted whenever the reference is unambiguous). The state vector $\mathbf{x}$ has dimensions $(2N+3) \times 1$ and consists of:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r}^T & \mathbf{p}_1^T & \mathbf{p}_2^T \ldots \mathbf{p}_N^T \end{bmatrix}^T \tag{1}$$

where $\mathbf{r} = [x_r \; y_r \; \phi_r]^T$ contains the position and orientation of the robot and $\mathbf{p}_i = [x_i \; y_i]^T$ denotes the position of the i^{th} landmark, all expressed with respect to a global frame of reference. The state covariance matrix P is a symmetric matrix of dimensions $(2N+3) \times (2N+3)$:

$$P = \begin{bmatrix} P_{rr} & P_{rp_1} & \cdot & \cdot & \cdot & P_{rp_N} \\ P_{p_1r} & P_{p_1p_1} & \cdot & \cdot & \cdot & P_{p_1p_N} \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ P_{p_Nr} & P_{p_Np_1} & \cdot & \cdot & \cdot & P_{p_Np_N} \end{bmatrix} \tag{2}$$

1) Propagation: The process model is given by:

$$\mathbf{x}_{k+1|k} = \mathbf{f}(\mathbf{x}_{k|k}, \mathbf{u}_k, \mathbf{w}_k) \tag{3}$$

where $\mathbf{u}_k = [v_k \; \omega_k]^T$ contains the measurements of the linear, v_k, and angular, ω_k, velocity of the robot at time-step k. $\mathbf{w}_k = [w_{v_k} \; w_{\omega_k}]^T$, with covariance Q_k, represents the noise in the linear and angular velocity measurement respectively.

The robot and landmarks' state estimates' propagation equations are:

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

$$\hat{\mathbf{p}}_{i_{k+1|k}} = \hat{\mathbf{p}}_{i_{k|k}} \tag{5}$$

where the map features are assumed to be stationary. The covariance propagation equation is given by:

$$P_{k+1|k} = \Phi_k P_{k|k} \Phi_k^T + G_k Q_k G_k^T \tag{6}$$

where

$$\Phi_k = \begin{bmatrix} \Phi_{rk} & 0_{3\times 2N} \\ 0^T_{3\times 2N} & I_{2N} \end{bmatrix}, G_k = \begin{bmatrix} G_{rk} \\ 0_{2N\times 2} \end{bmatrix} \tag{7}$$

$\Phi_{rk} = \nabla_{\mathbf{x}_r}(\mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, 0))$, $G_{rk} = \nabla_{\mathbf{w}}(\mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, 0))$ and I_{2N} is the $2N \times 2N$ identity matrix.

2) Update: The measurement model for a feature observation is:

$$\mathbf{z}_{k+1} = \mathbf{h}(\mathbf{x}_{k+1|k}) + \mathbf{n}_{k+1} \tag{8}$$

where $\mathbf{n}_{k+1} = [n_{d_{k+1}} \; n_{\theta_{k+1}}]^T$, with covariance R_{k+1}, denotes the noise in the range and bearing measurements, respectively, at time-step $k+1$.

The $2 \times (2N+3)$ measurement matrix is computed as:

$$H_{k+1} = [H_r \; 0_{2\times 2(i-1)} \; H_i \; 0_{2\times 2(N-i)}] \tag{9}$$

where $H_r = \nabla_{\mathbf{x}_r}(\mathbf{h}(\hat{\mathbf{x}}_{k+1|k}))$ and $H_i = \nabla_{\mathbf{p}_i}(\mathbf{h}(\hat{\mathbf{x}}_{k+1|k}))$.

Given the feature measurement, the state and covariance are updated as follows:

$$\mathbf{r}_{k+1} = \mathbf{z}_{k+1} - \mathbf{h}(\hat{\mathbf{x}}_{k+1|k}) \tag{10}$$

$$S_{k+1} = H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1} \tag{11}$$

$$K_{k+1} = P_{k+1|k} H_{k+1}^T S_{k+1}^{-1} \tag{12}$$

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

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1} S_{k+1} K_{k+1}^T \tag{14}$$

The dimensions of the Kalman gain matrix (cf. Eq. (12)) are $(2N+3) \times 2$. Thus, updating the covariance matrix (cf. Eq. (14)) has computational complexity $O(N^2)$, which for large number of landmarks, makes the real-time implementation of EKF-based SLAM impossible. The next section describes the details of the Global-Map Postponement approach that has computational cost $O(N)$.

B. Global-Map Postponement SLAM

Contrary to the approaches of [2], [13] that support postponement only when the robot operates within small areas (sub-maps), we hereafter present our method for postponement, Global-Map Postponement, that poses no restrictions on the motion of the robot. This discussion is necessary since our Power-SLAM algorithm is built on this general formulation of the delayed covariance update. Our goal in this section is to demonstrate that within this framework EKF-based SLAM requires $O(N)$ operations.

Consider the case where at time-step $k+1$, a new feature observation is processed to update the covariance. Equation (14) can be written as:

$$P_{k+1|k+1} = P_{k+1|k} - (K_{k+1} S_{k+1}^{1/2})(K_{k+1} S_{k+1}^{1/2})^T \tag{15}$$

where $S^{1/2}$ is a lower-triangular matrix obtained from the Cholesky factorization of the 2×2 matrix S. $(K_{k+1} S_{k+1}^{1/2})$ has dimensions $(2N+3) \times 2$ and can be split into two $(2N+3) \times 1$ vectors, $\mathbf{k}_1$, $\mathbf{k}_2$, to give:

$$P_{k+1|k+1} = P_{k+1|k} - \sum_{i=1}^{2} \mathbf{k}_i \mathbf{k}_i^T \tag{16}$$

In our approach the vector outer-product sum $\sum_{i=1}^{2} \mathbf{k}_i \mathbf{k}_i^T$ is never computed. Instead the $(2N+3) \times 1$ Kalman vectors $\mathbf{k}_i$ are stored for later processing.[3]

Repeating this process for all subsequent updates, at time-step $k+m$, the covariance update equation becomes:[4]

$$P_{k+m|k+m} = P_{k+m|k} - \sum_{i=1}^{m} \mathbf{k}_i \mathbf{k}_i^T \tag{17}$$

Maintaining this structure and for $m \ll N$, we hereafter show that both propagation and update can be performed with computational cost at most $O(mN)$.

1) Propagation: The covariance propagation equation at the next time step becomes (cf. Eqs. (6), (17)):

$$\begin{aligned} P_{k+m+1|k+m} &= \Phi_{k+m} P_{k+m|k} \Phi_{k+m}^T + G_{k+m} Q_{k+m} G_{k+m}^T \\ &\quad - \sum_{i=1}^{m} (\Phi_{k+m}\mathbf{k}_i)(\Phi_{k+m}\mathbf{k}_i)^T \\ &= P_{k+m+1|k} - \sum_{i=1}^{m} \mathbf{k}_i^* \mathbf{k}_i^{*T} \end{aligned} \tag{18}$$

where $P_{k+m+1|k}$, computed in linear time as in the standard SLAM, is the propagated covariance at time-step $k+m+1$ given measurements up to time-step k. Due to the structure of the matrix Φ_{k+m} (cf. Eq. (7)), calculation of the $\mathbf{k}_i^*$'s requires constant, $O(m)$, time.

2) Update: The residual covariance is given by (cf. Eqs. (11), (18)):[5]

$$\begin{aligned} S_{k+m+1} &= H_{k+m+1} P_{k+m+1|k} H_{k+m+1}^T + R_{k+m+1} \\ &\quad - \sum_{i=1}^{m} (H_{k+m+1}\mathbf{k}_i)(H_{k+m+1}\mathbf{k}_i)^T \end{aligned} \tag{19}$$

Since H_{k+m+1} contains only 2 non-zero blocks (cf. Eq. (9)), each term $H_{k+m+1}\mathbf{k}_i$ is calculated in constant time resulting in a total additional cost of $O(m)$; the remaining terms,

[3]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.

[4]Without loss of generality and in order to simplify the notation, assume that only 1 vector $\mathbf{k}_i$ is generated during each update step.

[5]After each modification of the $\mathbf{k}_i$ vectors (e.g., as during propagation), and in order to simplify the notation, we rename the resulting $\mathbf{k}_i^*$'s as $\mathbf{k}_i$'s.

$H_{k+m+1}P_{k+m+1|k}H_{k+m+1}^T + R_{k+m+1}$ are calculated in constant time as in standard SLAM.

The Kalman gain is expressed as (cf. Eqs. (12), (18)):

$$K_{k+m+1} = P_{k+m+1|k}H_{k+m+1}^T S_{k+m+1}^{-1} - \sum_{i=1}^{m} \mathbf{k}_i (H_{k+m+1}\mathbf{k}_i)^T S_{k+m+1}^{-1}$$

Note that since the terms $H_{k+m+1}\mathbf{k}_i$ have already been calculated (cf. Eq. (19)), the cost of computing $\mathbf{k}_i(H_{k+m+1}\mathbf{k}_i)^T S_{k+m+1}^{-1}$ is $O(N)$. Thus the summation term is evaluated in $O(mN)$. Similar to standard SLAM, the first term, $P_{k+m+1|k}H_{k+m+1}^T S_{k+m+1}^{-1}$, is calculated in $O(N)$.

Once the Kalman gain K_{k+m+1} is available, the state update (cf. Eq. (13)) is carried out in linear time. Finally, the covariance update equation is (cf. Eqs. (14), (18)):

$$P_{k+m+1|k+m+1} = P_{k+m+1|k} - \sum_{i=1}^{m+1} \mathbf{k}_i \mathbf{k}_i^T \tag{20}$$

where $\mathbf{k}_{m+1} = K_{k+m+1}S_{k+m+1}^{1/2}$.

3) Landmark Initialization: Every time a new landmark is detected, an estimate for this, $\hat{\mathbf{p}}_{N+1}$, is appended to the state vector. Furthermore, the covariance matrix $P_{k+m|k+m}$ (cf. Eq. (17)) needs to be modified. During this process, zeros are appended as the last two elements of the vectors $\mathbf{k}_i$, and the matrix $P_{k+m|k}$ is augmented to include the block matrices that correspond to (cf. Eq. (2)):[6] (i) the new landmark's covariance $P_{p_{N+1}p_{N+1}} = H_{N+1}^T \left(H_r P_{rr} H_r^T + R\right) H_{N+1}$ and (ii) the cross-correlation terms with the robot $P_{rp_{N+1}} = -P_{rr}H_r^T H_{N+1}$ and the rest of the landmarks $P_{p_i p_{N+1}} = -P_{p_i r}H_r^T H_{N+1},\ i = 1\ldots N$, where H_r and H_{N+1} are the measurement matrix blocks (cf. Eq. (9)).

Although $N+1$ terms need to be determined, the calculation of each of them has constant cost once P_{rr} and $P_{p_i r}$ are retrieved (cf. Eq. (17)) at $O(m)$ cost (note that P_{rr} and $P_{p_i r}$ are also needed for data association and can be determined by the same process). For example:

$$P_{rr}(k+m|k+m) = P_{rr}(k+m|k) - \sum_{i=1}^{m} \mathbf{k}_{ri}\mathbf{k}_{ri}^T$$

where $\mathbf{k}_{ri}$ denotes the first 3 elements of the vector $\mathbf{k}_i$ that correspond to the robot. Hence inserting new landmarks in the map requires $O(mN)$ operations.

Since no approximation has been made up to this point, the Global-Map Postponement SLAM will produce the exact same estimates as the standard SLAM, in *linear* time, for as long as the number of delayed updates, m, is significantly smaller than N. Inevitably, as the robot navigates and makes new observations, m will continuously increase. In order to maintain the structure of the covariance matrix (cf. Eq. (17)) while allowing for linear-time updates, the number, m, of $\mathbf{k_i}$ vectors in the vector outer-product sum $\sum_{i=1}^{m} \mathbf{k}_i\mathbf{k}_i^T$ (right-hand side of Eq. (17)) must be upper-bounded by $M_{max} \ll N$. This is achieved through the low-rank approximation described in the following section.

[6]Time indices have been dropped temporarily for clarity.

C. Low-Rank Approximation

Once $m = M_{max}$, the Power-SLAM algorithm employs a low-rank approximation of the accumulated, rank-M_{max} matrix[7]

$$D = \sum_{i=1}^{M_{max}} \mathbf{k}_i\mathbf{k}_i^T = \sum_{i=1}^{M_{max}} \lambda_i \mathbf{v}_i\mathbf{v}_i^T$$
$$\simeq \sum_{i=1}^{M_{min}} \lambda_i \mathbf{v}_i\mathbf{v}_i^T = \sum_{i=1}^{M_{min}} \mathbf{k}_i^*\mathbf{k}_i^{*T} = D^* \tag{21}$$

where $\lambda_1 > \lambda_2 \geq \lambda_3 \ldots \geq \lambda_{M_{max}}$ are the eigenvalues of D, $\mathbf{v}_i$, $i = 1\ldots M_{max}$ are the corresponding eigenvectors, and $\mathbf{k}_i^* = \sqrt{\lambda_i}\mathbf{v}_i$ are the new Kalman vectors. 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 D. Although 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 updates (cf. Eq. (17)) and hence the computation cost will remain linear, this is well justified for the following two reasons: (i) Most of the elements of the vectors $\mathbf{k}_i$ have very small values, except the ones that correspond to the robot, and those of the landmarks that are strongly correlated with the ones detected over the last $M_{max} - M_{min}$ time steps, and (ii) The rank-2 covariance update process (described in the following section), sparsifies the $\mathbf{k}_i$'s by replacing the highest, in the absolute value, of their elements with zeros. Hence, only few directions, $\mathbf{v}_i$, of 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 [11] is also based on the first reason mentioned above. In that case, however, *almost all* elements of $\mathbf{k}_i$ are discarded at every step, while only those corresponding to the robot and the observed landmark are retained. Due to this crude approximation and since in dense maps there exist strong correlations between neighborhoods of landmarks, the SWKF produces very conservative updates. Furthermore, and in stark contrast to the one-step approximations involved in [11] and [10], by employing the Global-Map Postponement 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, and thus significantly reduce the information loss.

A simplistic and very fast solution to the low-rank approximation described in Eq. (21) would be to select and retain 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 consistency of the estimator, it is not optimal unless: (a) $\mathbf{k}_i = \sqrt{\lambda_i}\mathbf{v}_i$ for $i = 1\ldots M_{min}$ or (b) $||\mathbf{k}_i|| \simeq 0$, for $i = M_{min}+1\ldots M_{max}$. While condition (a) is rarely satisfied in practice, condition (b) is usually true only for $i = M_{mid}+1\ldots M_{max}$, where

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

$M_{mid} \gg M_{min}$. Since the objective of the Power-SLAM estimator is to minimize the trace of the covariance matrix $P_{k+m|k+m}$ (cf. Eq. (17)), it is necessary to maximize the trace of the approximated Kalman vector outer-product sum D^* (cf. Eq. (21)). This requires us to determine the eigenvectors $\mathbf{v}_i$ that correspond to the M_{min} largest eigenvalues of D and it is accomplished by employing the Power Method [9].

In order to evaluate the computational cost of the Power Method, we hereafter briefly review its basic steps.

- Starting off with a random[8] vector $\mathbf{s}_0$, compute $\mathbf{s}_1 = D\mathbf{s}_0 = (\sum_{i=1}^{M_{max}} \mathbf{k}_i \mathbf{k}_i{}^T)\mathbf{s}_0 = \sum_{i=1}^{M_{max}} \mathbf{k}_i(\mathbf{k}_i^T \mathbf{s}_0)$. The cost of this step is $O(M_{max}N)$.
- Find the element of vector $\mathbf{s}_1$, $\mathbf{s}_1(\ell)$, with the largest absolute value. Set $\alpha = \mathbf{s}_1(\ell)$ and $\mathbf{s}_1 = \mathbf{s}_1/\alpha$.
- Repeat the above two steps n_p times[9] to acquire α (dominant eigenvalue λ_1) and $\mathbf{s}_{n_p}$ (dominant eigenvector $\mathbf{v}_1$). The cost of these iterations is $O(n_p M_{max}N)$.
- Setting $D = D - \lambda_1 \mathbf{v}_1 \mathbf{v}_1{}^T$ repeat the above process to acquire the second largest eigenvalue-eigenvector pair with cost $O(n_p(M_{max}+1)N)$.

This procedure can be repeated M_{min} times to determine all $(\lambda_i,\ \mathbf{v}_i),\ i = 1 \ldots M_{min}$ pairs for a total cost of $O(n_p M_{min}(M_{max} + \frac{M_{min}-1}{2})N) \approx O(n_p M_{min} M_{max} N)$ since $M_{min} \ll M_{max}$. Hence, for $n_p M_{min} M_{max} \ll N$, the computation cost of the Power Method is linear in N.

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 (b) mentioned earlier. Since $||\mathbf{k}_i|| \simeq 0$, for $i = M_{mid}+1 \ldots M_{max}$ where $M_{min} \ll M_{mid} \ll M_{max}$, the matrix D is first approximated as

$$D = \sum_{i=1}^{M_{max}} \mathbf{k}_i \mathbf{k}_i^T \simeq \sum_{i=1}^{M_{mid}} \mathbf{k}_i \mathbf{k}_i^T = \tilde{D} \tag{22}$$

Then the Power Method is applied to $\tilde{D}$ to determine its M_{min} largest eigenvectors and eigenvalues, i.e.,

$$\tilde{D} = \sum_{i=1}^{M_{mid}} \tilde{\lambda}_i \tilde{\mathbf{v}}_i \tilde{\mathbf{v}}_i^T \simeq \sum_{i=1}^{M_{min}} \tilde{\lambda}_i \tilde{\mathbf{v}}_i \tilde{\mathbf{v}}_i^T = \sum_{i=1}^{M_{min}} \tilde{\mathbf{k}}_i^* \tilde{\mathbf{k}}_i^{*T} = \tilde{D}^* \tag{23}$$

where $\tilde{\lambda}_i$, $\tilde{\mathbf{v}}_i$, and $\tilde{\mathbf{k}}_i^*$ are defined as in Eq. (21).

Selecting the M_{mid} largest (in the 2-norm) $\mathbf{k}_i$ vectors has cost $O(M_{max}N)$ for determining their magnitude and $O(M_{max}\log(M_{max}))$ for sorting them in ascending order. After this process is complete, the cost of the Power Method reduces to $O(n_p M_{min} M_{mid} N)$. Typical values of these parameters used in our tests are: (i) $M_{max} = (2-10)\%N$, (ii) $M_{mid} = \max\{2, (5-10)\%M_{max}\}$ (i.e., the Power Method is not used when $M_{mid} = 2$ which corresponds to $N < 250$, i.e, $\sim$ 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). These values can be adjusted on-line to meet the availability of computational resources.

2) Estimator Consistency: A key advantage of the presented low-rank approximation is that the covariance matrix remains conservative. Since (cf. Eq. (21)) $D \succeq D^*$, it is (cf. Eq.(17)):

$$P_{k+m|k+m} = P_{k+m|k} - D \ \succeq \ P_{k+m|k} - D^* = P^*_{k+m|k+m} \tag{24}$$

where $P^*_{k+m|k+m} = P_{k+m|k} - \sum_{i=1}^{M_{min}} \mathbf{k}_i^* \mathbf{k}_i^{*T}$ is now the new covariance. The same is true if $\tilde{D}^*$ is used instead of D^*, since (cf. Eqs. (22), (23)) $D \succeq \tilde{D} \succeq \tilde{D}^*$.

3) Quantifying the Information Loss: A concrete measure of the approximation involved is necessary in order to allow for adjusting the parameters M_{min} and M_{max} online to meet performance requirements. This can be done with complexity $O((M_{max}+M_{min})N)$ by computing the ratio $(\mathrm{tr}(D) - \mathrm{tr}(D^*))/\mathrm{tr}(D)$, where $\mathrm{tr}(D) = \sum_{i=1}^{M_{max}} \mathbf{k}_i{}^T \mathbf{k}_i$ and $\mathrm{tr}(D^*) = \sum_{i=1}^{M_{min}} \mathbf{k}_i^{*T} \mathbf{k}_i^*$.

D. Linear-Time, Rank-2 Covariance Updates

The main drawback of any low-rank approximation of D (cf. Eq. (21)) is that it does not guarantee loss of rank for $P_{k+m|k+m}$ (cf. Eq. (24)) after "infinite" time, which is expected when the system reaches steady-state [5], [16]. The reason for this is that matrix $P_{k+m|k}$ has rank $2N+3$, in general, while D has rank at most M_{max}; hence $P_{k+m|k+m} = P_{k+m|k} - D$ will have rank at least $2N+3-M_{max}$ with $M_{max} \ll N$. Furthermore, due to the propagation steps (cf. Eq. (18)), $P_{k+m|k}$ will, in general, continuously increase. The same is true for $D = \sum_{i=1}^{m} \mathbf{k}_i \mathbf{k}_i{}^T$, over the same period of time when the same landmarks are re-observed. In order to guarantee that the trace of $P_{k+m|k}$ will monotonically decrease, we require that certain elements of D are subtracted from $P_{k+m|k}$ at every time step. Note that any modification of D requires that it remains positive semi-definite, else the low-rank approximation described in the previous section will not guarantee consistency. In this work, we propose the following rank-2 covariance updates:

$$\begin{aligned} P_{k+m|k+m} &= P_{k+m|k} - \sum_{i=1}^{m} \mathbf{k}_i \mathbf{k}_i{}^T \\ &= \left(P_{k+m|k} - \delta P_j\right) - \left(\mathbf{k}_j^* \mathbf{k}_j^{*T} + \sum_{i=1, i\neq j}^{m} \mathbf{k}_i \mathbf{k}_i{}^T\right) \\ &= P_{k+m|k+1} - \sum_{i=1}^{m} \mathbf{k}_i^* \mathbf{k}_i^{*T} \end{aligned} \tag{25}$$

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

$$\delta P_j = (A_j \mathbf{k}_j)(A_j \mathbf{k}_j)^T + \mathbf{k}_j^* (A_j \mathbf{k}_j)^T + (A_j \mathbf{k}_j)\mathbf{k}_j^{*T} \tag{26}$$

and $\mathbf{k}_j \mathbf{k}_j{}^T = \delta P_j + \mathbf{k}_j^* \mathbf{k}_j^{*T}$ (the new matrix $D^* = \sum_{i=1}^{m} \mathbf{k}_i^* \mathbf{k}_i^{*T}$ remains positive semi-definite since it is still expressed as the accumulated sum of vector outer products). In the expressions above, $\mathbf{k}_j$ and A_j are a vector and a matrix that

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

[9] 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 $|1 - \mathbf{s}_{n_p}^T \mathbf{s}_{n_p-1}| < 10^{-6}$, i.e, the angle between these two vectors is smaller than $\sim 10^{-6}$ rad.

need to be determined so as the following two conditions are met:

(C1) δP_j (cf. Eq. (26)) is computed at a minimum cost, with at most $O(N)$ operations.

(C2) The trace of $P_{k+m|k+1} = P_{k+m|k} - \delta P_j$ is minimized (note that minimization of $\mathrm{tr}(P_{k+m|k+1})$ ensures minimization of $\mathrm{tr}(P_{k+m|k+m})$ when the vectors $\mathbf{k}_i$ are discarded).

Since the vector $\mathbf{k}_j$ can, in general, be dense, when computing $A_j\mathbf{k}_j$, (C1) requires that A_j has at most $n \ll N^2$ non-zero elements.[10] If these are distributed among $1 \leq p \leq N$ rows of A_j, then the cost for computing $A_j\mathbf{k}_j$ is n and this vector will have p non-zero elements. Therefore the cost for computing $(A_j\mathbf{k}_j)(A_j\mathbf{k}_j)^T$ is $\frac{p(p+1)}{2}$, since it is a symmetric matrix. When computing $\mathbf{k}_j^* = (I - A_j)\mathbf{k}_j = \mathbf{k}_j - A_j\mathbf{k}_j$, 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 A_j). Similarly, if $\mathbf{k}_j^*$ contains, $p-d$ zeros, computing $\mathbf{k}_j^*(A_j\mathbf{k}_j)^T$ requires $(N-(p-d))p$ operations. Hence the total cost becomes $c(p,d,n,N) = \frac{1}{2}\left(-p^2 + (2N+2d+1)p + 2(n+d)\right)$. This is a concave function of p that has maximum at $p = \frac{2N+2d+1}{2} > N$. Thus, it is monotonically increasing within the interval of interest $[1\ldots N]$ with minimum at $p = 1$. In that case, the total cost becomes $c(1,d,n,N) = N + n + d$, where, since $p = 1$, $d = 0$ or 1. Also, note that the matrix A_j contains only 1 row, e.g., the ξ-th, with $n \leq N$ non-zero elements, i.e.,

$$A_j^T = \left[\mathbf{0}\ldots\mathbf{a}_\xi\ldots\mathbf{0}\right] \tag{27}$$

We now turn our attention to (C2). Minimization of the trace of $P_{k+m|k+1}$ is equivalent to maximization of the trace of δP_j. Substituting Eq. (27) in Eq. (26), we have:

$$\begin{aligned}\mathrm{tr}(\delta P_j) &= \mathbf{k}_j^T\left(A_j + A_j^T - A_j^T A_j\right)\mathbf{k}_j \\ &= -(\mathbf{a}_\xi^T\mathbf{k}_j)^2 + 2k_{\xi j}(\mathbf{a}_\xi^T\mathbf{k}_j)\end{aligned} \tag{28}$$

which is a concave function of $\mathbf{a}_\xi$. Note that $k_{\xi j}$ is the ξ-th scalar element of vector $\mathbf{k}_j$. Computing the derivatives with respect to the elements of $\mathbf{a}_\xi$, the maximum of $\mathrm{tr}(\delta P_j)$, is reached when

$$(\mathbf{a}_\xi^T\mathbf{k}_j)\mathbf{k}_j = k_{\xi j}\mathbf{k}_j \tag{29}$$

This is trivially achieved by setting $\mathbf{a}_\xi = \mathbf{e}_\xi$, where $\mathbf{e}_\xi$ is the ξ-th canonical unit vector. In this case, A_j becomes a matrix of zeros, except the ξ-th diagonal element which is equal to one. Thus, $\mathbf{k}_j^*$ has the same elements as $\mathbf{k}_j$ except the ξ-th which is zero, and δP_j will have non-zero elements only in the ξ-th row and column (cf. Eq. (26)). Hence, the total cost for computing δP_j becomes $c(1,0,0,N) = N$. Subtracting δP_j from $P_{k+m|k+m}$ will also have cost N.

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 Eq. (28), we have $\max(\mathrm{tr}(\delta P_j)) = k_{\xi j}^2$. Hence maximization of $\mathrm{tr}(\delta P_j)$ is guaranteed by selecting among the $\mathbf{k}_j$ vectors, the one which has the maximum, in the absolute value, element $k_{\xi j}$. This can be determined at a cost of $O(mN)$.

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 $P_{k+m|k+1}$ and speed up convergence. Finally, we should note that the resulting matrix δP_j can be easily shown to have rank 2 (hence the name).

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

IV. Simulations

A. Simulation Setup

The simulation setup used to validate the performance of the Power- SLAM algorithm has been implemented in MATLAB. The robot starts at a known position and follows the 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 std. dev. $\sigma_v = 3\%v$, and $\sigma_\omega = 3\%\omega$, (ii) range d, with $\sigma_d = 8$ cm, and (iii) bearing θ with $\sigma_\theta = 1$ degree.

In this experiment, the robot observes about 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. 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 ($M_{min} = 1$).

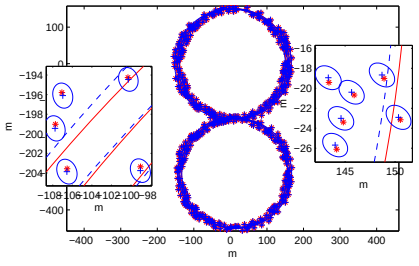

Fig. 1. Actual robot trajectory (solid red line), actual landmark positions (*), Power-SLAM estimated robot trajectory (dashed blue line), Power-SLAM estimated landmark positions (+), and their 3σ uncertainty ellipsoids. Insets are zoomed sections for better viewing of the uncertainty ellipsoids.

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 [11], and (iii) CKF SLAM [10]. 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

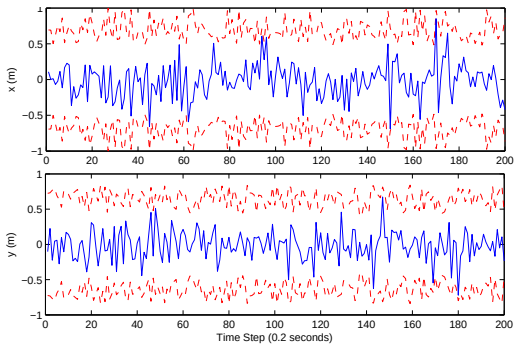

Fig. 2. Measurement residuals (solid) and $3-\sigma$ bounds (dashed).

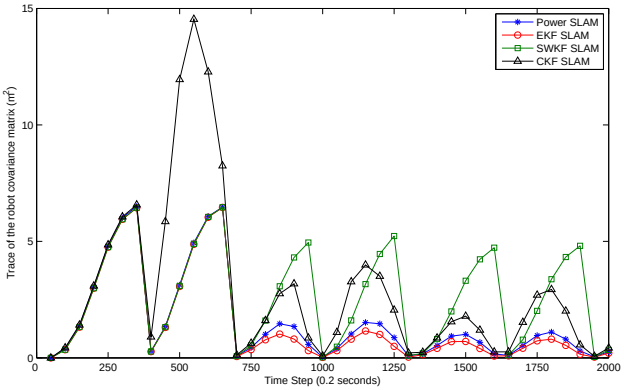

Fig. 3. Trace of the robot covariance matrix.

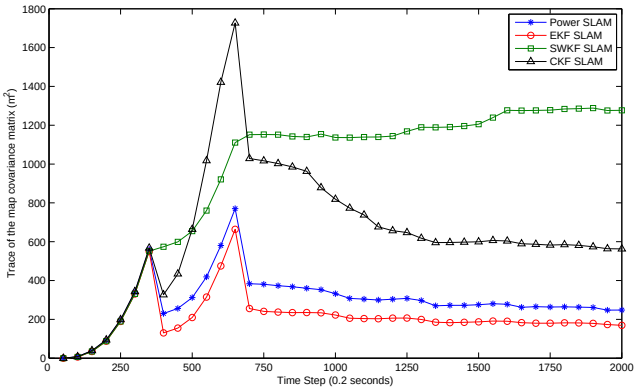

Fig. 4. Trace of the map covariance matrix.

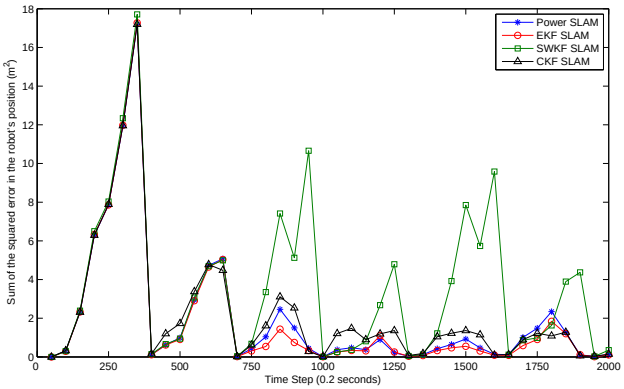

Fig. 5. Sum of the squared error in the robot's position.

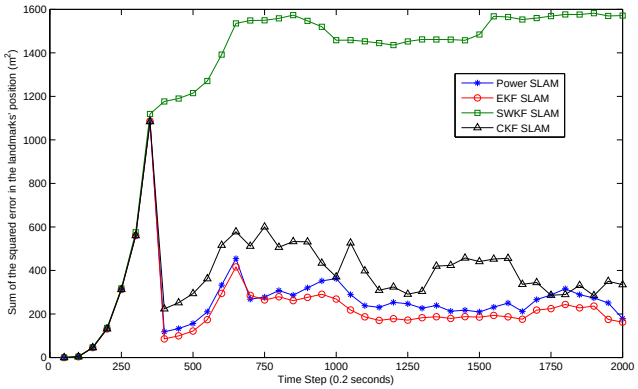

Fig. 6. Sum of the squared error in the landmarks' position.

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 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 $3-\sigma$ ellipsoids of uncertainty for the estimated landmark positions contain the true positions, indicating consistency. Fig. 2 depicts the measurement residuals along with the $3-\sigma$ 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, which was expected based on the analysis of Section III-C.2.

We now turn our attention to the quantitative results presented in Figs. 3-6. Although all 3 linear-complexity estimators are consistent, 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 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 by (i) delaying approximations over large time horizons, and (ii) extracting and retaining the most informative Kalman vectors during each approximation, is able to minimize the information loss.

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 (Fig. 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. Finally, we should note that although additional results cannot be presented here due to space limitations, the Power-SLAM algorithm has been extensively tested for different values of the parameters M_{min}, M_{mid}, M_{max} (adjusted to meet processing availability), and has consistently outperformed both the SWKF and the CKF estimators.

V. 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 mapped. 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. 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 even for the exact same number of operations. Extensive simulation 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.

VI. Acknowledgements

This work was supported by the University of Minnesota (DTC), the NASA Mars Technology Program (MTP-1263201), and the National Science Foundation (EIA-0324864, IIS-0643680).

References

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

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

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

[4] G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (slam) problem. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1009–1014, San Francisco, Apr. 2000.

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

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

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

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

[9] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

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

[11] S.J. Julier. A sparse weight kalman filter approach to simultaneous localisation and map building. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1251–1256, Oct 2001.

[12] S.J. Julier and J.K. Uhlmann. Simultaneous localisation and map building using split covariance intersection. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1257–1262, Oct 2001.

[13] J. Knight, A. Davison, and I. Reid. Towards constant time slam using postponement. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 405–413, Oct. 2001.

[14] J. Leonard and H.J.S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Proc. 9th Int. Symp. Robot. Res.*, pages 316–321, UT, Oct. 1999.

[15] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: a factored solution to the simultaneous localization and mapping problem. In *Eighteenth National Conf. on Artificial Intelligence*, pages 593–598, Menlo Park, CA, 2002.

[16] A.I. Mourikis and S.I Roumeliotis. Analytical characterization of the accuracy of slam without absolute orientation measurements. In *Proc. Robotics: Science and Systems Conf.*, Philadelphia, PA, Aug. 2006.

[17] P. Moutarlier and R. Chatila. Stochastic multi-sensory data fusion for mobile robot location and environmental modeling. In *Fifth Symp. Robot. Res.*, pages 85–94, Tokyo, 1989.

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

[19] M. A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *Proc. of the Eighteenth Int. Joint Conf. on Artificial Intelligence*, pages 1157–1164, San Francisco, 2003.

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

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

[22] S.B. Williams, G. Dissanayake, and H. Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 406–411, May 2002.