A Constant-Time Algorithm for Vector Field SLAM Using an Exactly Sparse Extended Information Filter

Jens-Steffen Gutmann, Ethan Eade, Philip Fong, and Mario Munich
Evolution Robotics, 1055 E. Colorado Blvd., Suite 410, Pasadena, CA 91106
{steffen,ethan,philip,mario}@evolution.com

***Abstract*— The constraints of a low-cost consumer product pose a major challenge for designing a localization system. In previous work, we introduced Vector Field SLAM [5], a system for simultaneously estimating robot pose and a vector field induced by stationary signal sources present in the environment. In this paper we show how this method can be realized on a low-cost embedded processing unit by applying the concepts of the Exactly Sparse Extended Information Filter [15]. By restricting the set of active features to the 4 nodes of the current cell, the size of the map becomes linear in the area explored by the robot while the time for updating the state can be held constant under certain approximations. We report results from running our method on an ARM 7 embedded board with 64 kByte RAM controlling a Roomba 510 vacuum cleaner in a standard test environment.**

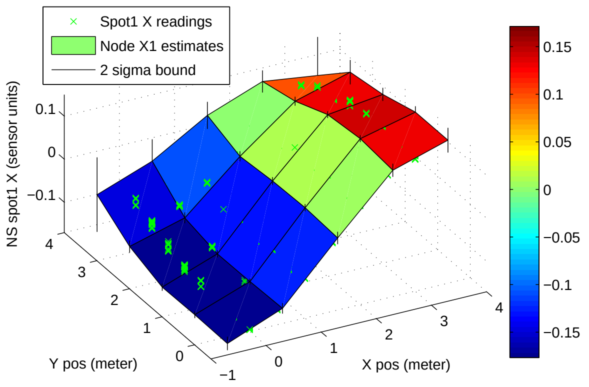

Fig. 1. Component *Spot1 X* of a vector field learned by EKF-SLAM.

I. Introduction

Deploying a localization system for the autonomous navigation of a low-cost consumer product is a non-trivial task. The majority of today's robotic floor cleaners employ relatively simple behaviors resulting in a more or less random movement in the environment. In order to achieve more systematic cleaning, the robot needs to estimate and track its position, a pre-requisite for mapping areas it has already cleaned and knowing where to go next.

Recently, a few notable systematic cleaners were introduced. Samsung's Hauzen vacuum cleaner, moving in straight lines while keeping its initial orientation, utilizes a SLAM system based on ceiling vision [9] for keeping track of its pose.

Neato Robotics uses a miniature laser range finder [11] on their XV-11 robot and employ standard range finder mapping techniques [10, 13] for estimating map and robot position.

Our solution to localization for a consumer product uses the information from time-stationary signals present in the environment. Examples of such signals are the signal strengths to WiFi stations or cellular networks, or the direction to unique beacons in the environment. These signals can often be well modelled when observed directly, i.e. sensor and signal source are on a direct path [4]. Signal reflections from walls and furniture, however, can disturb the measurements significantly.

Trying to model or discard signal reflections is a difficult task often resulting in unreliable and error-prone position estimates. Instead, recent research focused on building representations that map ground positions to expected signal values as they would appear at those locations. If learned off-line in a training phase, such a representation can practically be used for localization [6]. Ferris *at al.* showed that using Gaussian Process Latent Variable Models allows building of such signal maps from collected WiFi signal strength data [1].

In our work, we represent this signal map as a vector field over space approximated by a piece-wise linear function. Our model contains estimates of the signal values at so-called *node* positions of a regular grid laid onto the ground. Signal values at arbitrary positions are predicted by bilinear interpolation. Using this model, both vector field and robot pose are then learned through the application of simultaneous localization and mapping (SLAM). Previously, we introduced this approach as *Vector Field SLAM* and showed how an extended Kalman filter (EKF) as well as a non-linear optimization can accurately localize the robot and estimate signal maps in an environment equipped with a pair of active beacons [5]. As an example, Fig. 1 shows one component of a vector field learned by our EKF implementation of Vector Field SLAM.

Since the time and space complexity of EKF-SLAM is quadratic in the number of map features [13], an implementation on low-cost hardware with limited processing and memory resources is not practical. The contribution of this paper is the development of a constant-time algorithm with linear space requirements that allows running Vector Field SLAM on such hardware. We achieve this by a formulation of the exactly sparse extended information filter (ESEIF) [15] in which the set of *active* features is limited to the signal values of the four nodes of the cell occupied by the robot. In experiments, we verify that our application of ESEIF produces pose estimates comparable to our previous EKF solution.

This paper is organized as follows. After an introduction to Vector Field SLAM in the next section, we briefly describe the

ESEIF in Section III. Section IV presents our application of the ESEIF to Vector Field SLAM followed by an implementation using active beacons in Section V. Experimental results are described in Section VI. We draw conclusions in Section VII.

II. Vector Field SLAM

Vector Field SLAM learns the signal distribution of a time-invariant vector field over the environment while at the same time tracking the pose of the robot. A vector field of dimension M is defined as

$$\mathrm{VF} : \mathrm{SE}(2) \quad \rightarrow \quad \mathbb{R}^M \tag{1}$$

mapping a ground pose to a vector of signal values.

We assume that the dependency of signals on robot orientation can be fully characterized by some internal calibration parameters $\mathbf{c}$ of the sensor that allow for a *rotational variability* of measurements. For example, a WiFi receiver might show changes in signal strength on rotation caused by the directional sensitivity of the antenna. A sensor measuring bearing and elevation to beacons can show variations due to calibration errors of the sensor's vertical axis.

Under these assumptions we decompose the space of poses $\mathrm{SE}(2)$ into position and orientation. The vector field over position is then modeled as a piece-wise linear function by laying a regular grid of node positions $\mathbf{b}_i = (b_{i,x}, b_{i,y})^T$, $i = 1 \ldots N$ onto the ground. This creates cells with one node at each of the cell's four corners. Each node i holds a vector $\mathbf{m}_i \in \mathbb{R}^M$ describing the expected signal values when the robot is located at $\mathbf{b}_i$ and oriented in a preset direction θ_0.

For an arbitrary robot position with orientation θ_0, the signal values are computed by bilinear interpolation from the four nodes of the cell containing the robot. Let $\mathbf{x}_t = (x, y, \theta)^T$ be the robot pose and $\mathbf{b}_{i_0} \ldots \mathbf{b}_{i_3}$ the cell nodes enclosing the robot as shown in Fig. 2. The signal values at (x, y) with orientation θ_0 are then computed as:

$$h_0(x, y, \mathbf{m}_1 \ldots \mathbf{m}_N) \quad = \quad \sum_{j=0}^{3} w_j \mathbf{m}_{i_j} \tag{2}$$

where $\mathbf{m}_{i_0} \ldots \mathbf{m}_{i_3}$ are the signal values at the cell nodes and $w_0 \ldots w_3$ weights of the bilinear interpolation:

$$w_0 = \frac{(b_{i_1,x} - x)(b_{i_2,y} - y)}{(b_{i_1,x} - b_{i_0,x})(b_{i_2,y} - b_{i_0,y})} \tag{3}$$

$$w_1 = \frac{(x - b_{i_0,x})(b_{i_2,y} - y)}{(b_{i_1,x} - b_{i_0,x})(b_{i_2,y} - b_{i_0,y})} \tag{4}$$

$$w_2 = \frac{(b_{i_1,x} - x)(y - b_{i_0,y})}{(b_{i_1,x} - b_{i_0,x})(b_{i_2,y} - b_{i_0,y})} \tag{5}$$

$$w_3 = \frac{(x - b_{i_0,x})(y - b_{i_0,y})}{(b_{i_1,x} - b_{i_0,x})(b_{i_2,y} - b_{i_0,y})}. \tag{6}$$

The final signal values are computed by taking into account robot orientation θ and sensor calibration $\mathbf{c}$:

$$h(\mathbf{x}_t, \mathbf{c}, \mathbf{m}_1 \ldots \mathbf{m}_N) = h_R(h_0(x, y, \mathbf{m}_1 \ldots \mathbf{m}_N), \theta, \mathbf{c}). \tag{7}$$

Here h_R is a continuous function that transforms the interpolated signal values obtained through (2) by the robot orientation and calibration. This is usually a rotation by orientation

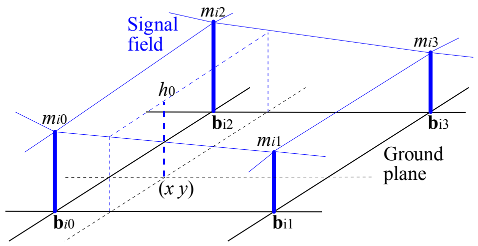

Fig. 2. Bilinear interpolation from cell nodes

θ followed by a correction with the rotational variability $\mathbf{c}$. We will provide a particular instance of h_R in an application using active beacons in Section V.

Robot path $\mathbf{x}_0 \ldots \mathbf{x}_T$, $\mathbf{x}_t \in \mathrm{SE}(2)$, rotational variability $\mathbf{c}$ and node values $\mathbf{m}_1 \ldots \mathbf{m}_N, \mathbf{m}_i \in \mathbb{R}^M$ are estimated through SLAM. Without loss of generality, $\mathbf{x}_0 = (0, 0, 0)^T$. At each time step $t = 1 \ldots T$ the robot receives a motion input $\mathbf{u}_t$ with covariance $\mathbf{R}_t$ and a measurement $\mathbf{z}_t$ with covariance $\mathbf{Q}_t$.

A motion model defined by a function g describes the motion of the robot since the previous time step:

$$\mathbf{x}_t \quad = \quad g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{e}_u \tag{8}$$

where $\mathbf{e}_u$ is a zero mean error with covariance $\mathbf{R}_t$.

Furthermore, our sensor model (7) predicts an observation given current robot pose, sensor calibration and features:

$$\mathbf{z}_t \quad = \quad h(\mathbf{x}_t, \mathbf{c}, \mathbf{m}_1 \ldots \mathbf{m}_N) + \mathbf{e}_z \tag{9}$$

where $\mathbf{e}_z$ is a zero mean error with covariance $\mathbf{Q}_t$.

In online SLAM [13], an algorithm estimates the state

$$\mathbf{y}_t \quad = \quad (\mathbf{x}_t, \mathbf{c}, \mathbf{m}_1 \ldots \mathbf{m}_N)^T \tag{10}$$

recursively over time. The EKF is one example of such an algorithm. Starting from an initial mean μ_0 and covariance Σ_0 the state is updated on motion and sensor observation using the well-known EKF equations [5, 12, 13].

In general, the initial state often contains a minimal set of map features. For Vector Field SLAM we include only the four nodes of the first cell in the initial state:

$$\mathbf{y}_0 \quad = \quad (\mathbf{x}_0, \mathbf{c}, \mathbf{m}_1 \ldots \mathbf{m}_4)^T. \tag{11}$$

An estimate for $\mathbf{y}_0$ can be obtained through non-linear optimization on a small set of collected measurements. In our approach we use RANSAC [2] for estimating a linear model of the vector field around the robot and then compute mean values $\hat{\mathbf{m}}_1 \ldots \hat{\mathbf{m}}_4$ of the four nodes. The initial mean and covariance are then found as

$$\mu_0 \quad = \quad (0, 0, 0, \hat{\mathbf{c}}, \hat{\mathbf{m}}_1 \ldots \hat{\mathbf{m}}_4)^T \tag{12}$$

$$\Sigma_0 \quad = \quad \mathrm{diag}(0, 0, 0, \infty \ldots \infty) \tag{13}$$

where $\hat{\mathbf{c}}$ is an initial guess of rotational variability. Although a tighter initial covariance can be found from the residual fitting error, in practice Eq. (13) is often sufficient since the node covariances decrease quickly after a few measurements.

During exploration new nodes need to be added to the state. In our approach these are created by a *structure copy*. Let $\mathbf{y}_t = (\mathbf{x}_t, \mathbf{c}, \mathbf{m_1} \ldots \mathbf{m}_n)$ be the state vector at time t and

$\mathbf{b}_{n+1}$ the position of a new node. Let $\mathbf{b}_{j_1}$ and $\mathbf{b}_{j_2}$ be the positions of two nodes already contained in the state that lie on a straight line passing through $\mathbf{b}_{n+1}$. The new node $\mathbf{m}_{n+1}$ is then initialized by extrapolation from $\mathbf{m}_{j_1}$ and $\mathbf{m}_{j_2}$ as:

$$\mathbf{m}_{n+1} \quad = \quad \mathbf{A}_1\mathbf{m}_{j_1} + \mathbf{A}_2\mathbf{m}_{j_2} + \mathbf{e}_1 \tag{14}$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ contain the extrapolation factors and $\mathbf{e}_1$ is a zero mean error term with a preset covariance $\mathbf{S}$ allowing the node values to change. An example of extrapolation factors is described in our implementation in Section V. The state of the EKF is then augmented with the new node and its covariance is expanded accordingly [5].

In our recent work we collected data on a robotic vacuum cleaner moving in an environment with active beacons. Our results demonstrate that both EKF-SLAM and an off-line GraphSLAM version of Vector Field SLAM significantly improve position estimates when compared to raw odometry or a direct triangulation of sensor data [5]. The limitations of EKF-SLAM are in the time and space complexity which is quadratic in the number of map features [13]. While this allows running the method on PC hardware, the requirements practically exceed the resources of low-cost embedded processors.

III. Exactly Sparse Extended Information Filter

In order to solve the problem of scalability in SLAM, representations of the state distribution different than the mean and covariance of an EKF have been researched. In a seminal paper, Thrun *et al.* [14] presented a sparse extended information filter (SEIF) with almost linear time complexity. Recently Walter *et al.* [15] further developed this approach to an exactly sparse extended information filter (ESEIF) producing estimates that are conservative relative to the EKF solution.

In general, the extended information filter (EIF) represents the state $\mathbf{y}_t$ at time t by an information vector η_t and information matrix Λ_t. The relation to the mean μ_t and covariance Σ_t of an EKF is given by:

$$\eta_t = \Sigma_t^{-1}\mu_t \qquad \Lambda_t = \Sigma_t^{-1}. \tag{15}$$

On sensor observation the EIF state is updated as [14]:

$$\begin{aligned} \eta_t &= \bar{\eta}_t + \mathbf{H}_y^T\mathbf{Q}_t^{-1}(\mathbf{z}_t - h(\bar{\mu}_t) + \mathbf{H}_y\bar{\mu}_t) & (16)\\ \Lambda_t &= \bar{\Lambda}_t + \mathbf{H}_y^T\mathbf{Q}_t^{-1}\mathbf{H}_y & (17) \end{aligned}$$

where $\bar{\eta}_t$ and $\bar{\Lambda}_t$ are the EIF state after a motion update (see below) and $\mathbf{H}_y$ is the Jacobian of h w.r.t. the state evaluated at an estimate $\bar{\mu}_t$ of the mean:

$$\mathbf{H}_y \quad = \quad \frac{\partial h}{\partial \mathbf{y}}(\bar{\mu}_t). \tag{18}$$

The update time is quadratic in the number of observed features which is often small in practical applications.

The EIF update on motion is more complicated. It can be formulated by first augmenting the state with the new robot pose followed by a marginalization over the old pose [15]. Given a robot pose estimate μ_x and the EIF state at time $t-1$ separated into robot variables $\mathbf{x}$ and map variables $\mathbf{M}$:

$$\eta_{t-1} = \begin{pmatrix} \eta_x \\ \eta_M \end{pmatrix} \qquad \Lambda_{t-1} = \begin{pmatrix} \Lambda_{xx} & \Lambda_{xM} \\ \Lambda_{Mx} & \Lambda_{MM} \end{pmatrix} \tag{19}$$

we compute information vector and matrix on motion as:

$$\bar{\eta}_t = \begin{pmatrix} \bar{\eta}_x \\ \bar{\eta}_M \end{pmatrix} \qquad \bar{\Lambda}_t = \begin{pmatrix} \bar{\Lambda}_{xx} & \bar{\Lambda}_{xM} \\ \bar{\Lambda}_{Mx} & \bar{\Lambda}_{MM} \end{pmatrix} \tag{20}$$

where

$$\begin{aligned} \bar{\eta}_x &= \Psi^T\eta_x + (\mathbf{R}_t + \mathbf{G}_x\Lambda_{xx}^{-1}\mathbf{G}_x^T)^{-1}\Delta_x & (21)\\ \bar{\eta}_M &= \eta_M - \Lambda_{Mx}(\Lambda_{xx}^{-1}\eta_x - \Omega\eta_x - \Psi\Delta_x) & (22)\\ \bar{\Lambda}_{xx} &= (\mathbf{R}_t + \mathbf{G}_x\Lambda_{xx}^{-1}\mathbf{G}_x^T)^{-1} & (23)\\ \bar{\Lambda}_{xM} &= \Psi^T\Lambda_{xM} \qquad \bar{\Lambda}_{Mx} \;=\; \Lambda_{Mx}\Psi & (24)\\ \bar{\Lambda}_{MM} &= \Lambda_{MM} - \Lambda_{Mx}(\Lambda_{xx}^{-1} - \Omega)\Lambda_{xM} & (25)\\ \Delta_x &= g(\mu_x, \mathbf{u}_t) - \mathbf{G}_x\mu_{\mathbf{x}} & (26)\\ \Psi &= \mathbf{G}_x^{-1} - \mathbf{G}_x^{-1}\mathbf{R_t}(\mathbf{R}_t + \mathbf{G}_x\Lambda_{xx}^{-1}\mathbf{G}_x^T)^{-1} & (27)\\ \Omega &= \Lambda_{xx}^{-1}\mathbf{G}_x^T(\mathbf{R}_t + \mathbf{G}_x\Lambda_{xx}^{-1}\mathbf{G}_x^T)^{-1}\mathbf{G}_x\Lambda_{xx}^{-1} & (28) \end{aligned}$$

and $\mathbf{G}_x$ is the Jacobian of motion model (8) w.r.t. robot pose:

$$\mathbf{G}_x \quad = \quad \frac{\partial g}{\partial \mathbf{x}}(\mu_x, \mathbf{u}_t). \tag{29}$$

Our formulation through Eqs. (20-29) is equivalent to the one presented by Walter *et al.* [15] after utilizing the Woodbury matrix identity [3]. The advantage of Eqs. (20-29) is that the motion covariance $\mathbf{R}_t$ itself and not its inverse is used for updating the state. This allows for motions with arbitrarily small errors. The price of this formulation is that the motion Jacobian $\mathbf{G}_x$ must be invertible. In Section V, we show that this holds for a standard odometry motion model [13].

The most expensive operation of the motion update lies in Eq. (25) and is quadratic in the number of non-zero elements of the cross-information matrix Λ_{xM}, i.e. the number of map features that share information with the robot pose. In its plain form the EIF does not restrict this number and thus, does not provide a better solution than the EKF in terms of scalability [15]. The advancements of the SEIF and ESEIF are in controlling this number such that the space and time complexities are bounded.

The ESEIF divides map features into *active* and *passive* ones based upon whether or not they share information with the robot pose. The algorithm then poses an upper bound Γ_a on the number N_a of active features. As long as $N_a \leq \Gamma_a$ the state is updated on motion and sensor observation using the EIF Eqs. (16-29) as presented above.

When receiving an observation $\mathbf{z}_t$ that would cause an EIF update (16,17) to exceed the Γ_a threshold, a sparsification procedure takes place. For this $\mathbf{z}_t$ is partitioned into two sets:

$$\mathbf{z}_t \quad = \quad \{\mathbf{z}_\alpha, \mathbf{z}_\beta\} \tag{30}$$

such that $|\mathbf{z}_\beta| \leq \Gamma_a$. The measurements $\mathbf{z}_\alpha$ are then used in a regular EIF update (16,17) followed by marginalizing over the robot pose and relocating it using the measurements $\mathbf{z}_\beta$.

Marginalization removes the robot variables from the state:

$$\begin{aligned} \check{\eta}_t &= \bar{\eta}_M - \bar{\Lambda}_{Mx}\bar{\Lambda}_{xx}^{-1}\bar{\eta}_x & (31)\\ \check{\Lambda}_t &= \bar{\Lambda}_{MM} - \bar{\Lambda}_{Mx}\bar{\Lambda}_{xx}^{-1}\bar{\Lambda}_{xM} & (32) \end{aligned}$$

and can be computed in time quadratic in Γ_a.

For relocating the robot, the ESEIF computes a new robot pose $\mathbf{x}_t$ given the map features $\mathbf{M}_\beta$ and measurements $\mathbf{z}_\beta$:

$$\mathbf{x}_t \quad = \quad f(\mathbf{M}_\beta, \mathbf{z}_\beta) + \mathbf{e}_x \tag{33}$$

where $\mathbf{e}_x$ is a zero mean error with covariance $\mathbf{R}_x$. The final state is then computed by augmenting the new robot pose:

$$\breve{\eta}_t \quad = \quad \begin{pmatrix} \mathbf{R}_x^{-1}\left(f(\breve{\mu}_{M_\beta}, \mathbf{z}_\beta) - \mathbf{F}_M \breve{\mu}_t\right) \\ \breve{\eta}_t - \mathbf{F}_M^T \mathbf{R}_x^{-1}\left(f(\breve{\mu}_{M_\beta}, \mathbf{z}_\beta) - \mathbf{F}_M \breve{\mu}_t\right) \end{pmatrix} \tag{34}$$

$$\breve{\Lambda}_t \quad = \quad \begin{pmatrix} \mathbf{R}_x^{-1} & -\mathbf{R}_x^{-1}\mathbf{F}_M \\ -\mathbf{F}_M^T \mathbf{R}_x^{-1} & \breve{\Lambda}_t + \mathbf{F}_M^T \mathbf{R}_x^{-1} \mathbf{F}_M \end{pmatrix} \tag{35}$$

where $\mathbf{F}_M$ is the Jacobian of relocation model (33) w.r.t. map:

$$\mathbf{F}_M \quad = \quad \frac{\partial f}{\partial \mathbf{M}}(\breve{\mu}_{M_\beta}, \mathbf{z}_\beta). \tag{36}$$

This requires time quadratic in $|\mathbf{z}_\beta|$ and resets the set of active features to those in $\mathbf{z}_\beta$. Therefore, since $|\mathbf{z}_\beta| \leq \Gamma_a$ the computation time is fully controlled by parameter Γ_a.

In the ESEIF, an estimate of the mean μ_t is needed when computing Jacobians or updating the information vector. This mean can be found by solving a linear equation system:

$$\Lambda_t \mu_t \quad = \quad \eta_t. \tag{37}$$

A naïve approach of inverting Λ_t requires time cubic in the size of the state which destroys the scalability of the algorithm. As Walter *et al.* [15] pointed out we are often only interested in a subset of the mean. Therefore, (37) is partitioned into

$$\begin{pmatrix} \Lambda_{ll} & \Lambda_{lb} \\ \Lambda_{bl} & \Lambda_{bb} \end{pmatrix} \begin{pmatrix} \mu_l \\ \mu_b \end{pmatrix} \quad = \quad \begin{pmatrix} \eta_l \\ \eta_b \end{pmatrix} \tag{38}$$

where μ_l is a *local portion* to solve for and μ_b the *benign portion* for which we already have an estimate. This leads to:

$$\mu_l \quad = \quad \Lambda_{ll}^{-1}(\eta_l - \Lambda_{lb}\mu_b) \tag{39}$$

and requires only a subset of μ_b corresponding to the non-zero elements in Λ_{lb}, i.e. the Markov blanket of the local portion. The time for computing (39) is linear in the size of the Markov blanket and cubic in the size of the local portion.

The ESEIF has been shown to produce estimates close to those of an EKF with significant savings in memory usage and run-time improving the scalability of on-line SLAM [15].

IV. Vector Field SLAM using ESEIF

We now have the tools ready for using the ESEIF in Vector Field SLAM. For the upper bound Γ_a of active features, we seek the smallest possible number while still preserving a good level of information necessary for approximating the full Gaussian distribution of EKF-SLAM. By analyzing the sensor model (7), we note that robot pose $\mathbf{x}_t$, sensor calibration $\mathbf{c}$, and the four nodes $\mathbf{m}_{i_0} \ldots \mathbf{m}_{i_3}$ enclosing the robot are sharing information between each other. Our goal is to restrict the number of active features to this configuration, thus $\Gamma_a = 4$.

As long as the robot stays within the same cell, the ESEIF state is updated on sensor observation and motion through EIF Eqs. (16-29) where the set $\mathbf{M}$ in (19,20) include sensor calibration $\mathbf{c}$ and all nodes $\mathbf{m}_1 \ldots \mathbf{m}_n$ of the current state $\mathbf{y}_t$.

After moving into another cell, we reset the set of active features to the four nodes of the new cell when receiving a new sensor measurement. This involves performing the sparsification procedure of the ESEIF. In our sensor model (7) an observation $\mathbf{z}_t$ is associated with only the nodes of one cell. It is therefore not possible to partition $\mathbf{z}_t$ into two non-empty sets corresponding to different map features. We then choose the partitioning of

$$\mathbf{z}_\alpha = \emptyset \qquad \mathbf{z}_\beta = \mathbf{z}_t \tag{40}$$

i.e. all measurements are used for relocating the robot.

The ESEIF sparsification applies the marginalization in Eqs. (31,32). In Vector Field SLAM we not only marginalize out the robot pose $\mathbf{x}_t$, but also the rotational variability $\mathbf{c}$. This has certain advantages which we outline further below. Thus, for applying Eqs. (31,32) we set the robot variables to $\mathbf{x} = \{\mathbf{x}_t, \mathbf{c}\}$ and map variables to $\mathbf{M} = \{\mathbf{m}_1 \ldots \mathbf{m}_n\}$.

For re-inserting robot pose and rotational variability, we need to find a function f for Eq. (33) that generates these from the map $\mathbf{M}_\beta$ and observation $\mathbf{z}_\beta$. Unfortunately this turns out to be a difficult task. In Vector Field SLAM there can be areas where a single observation might not provide full information about the robot pose, e.g. when only a small subset of signals are observed, or when the measurement noise is large. Furthermore, for determining rotational variability, often measurements from different robot orientations are needed.

Instead of following Eqs. (34,35) we propose a two-step procedure for relocating robot pose and rotational variability. First, we augment the state with blank robot variables

$$\breve{\eta}_t = \begin{pmatrix} \breve{\Lambda}_{xx}\bar{\mu}_x \\ \breve{\eta}_t \end{pmatrix} \qquad \breve{\Lambda}_t = \begin{pmatrix} \breve{\Lambda}_{xx} & 0 \\ 0 & \breve{\Lambda}_t \end{pmatrix} \tag{41}$$

where $\bar{\mu}_x$ is the mean of robot variables before marginalization, and $\breve{\Lambda}_{xx}$ is a prior on robot pose and sensor calibration.

In the second step, the measurements $\mathbf{z}_t$ are used for updating this state using the regular EIF Eqs. (16,17). This creates new information between robot pose, sensor calibration and the nodes of the new cell, while all other cross information between robot variables and map are zero, thus $N_a \leq \Gamma_a$.

The prior $\breve{\Lambda}_{xx}$ plays an important role in our relocation procedure. For a truly conservative filter, the prior should be set to $\breve{\Lambda}_{xx} = 0$. The formulation then becomes equivalent to Eqs. (34,35) modulo different approximation errors in the involved Jacobians. In practical applications, however, it is often useful to restrict robot and sensor calibration to an area around the estimate before sparsification, since there was no physical cause that would have allowed them to change by a large amount. Since a non-zero prior can cause the filter to become over-confident, it has to be carefully chosen. In our application we maintain an estimate of the covariance Σ_{xx} of robot variables and compute $\breve{\Lambda}_{xx}$ as

$$\breve{\Lambda}_{xx} \quad = \quad (\Sigma_{xx} + \mathbf{R}_0)^{-1} \tag{42}$$

where $\mathbf{R}_0$ is a minimal covariance that needs to be tuned in order to ensure the filter produces non-optimistic estimates.

```
Motion update:
$(\eta_{t-1},\Lambda_{t-1}) \xrightarrow[\mathbf{x}=\{\mathbf{x}_t\},\mathbf{M}=\{\mathbf{c},\mathbf{m}_1\ldots\mathbf{m}_n\}]{(19\text{-}29)} (\bar{\eta}_t,\bar{\Lambda}_t)$
Sensor observation update:
if robot in same cell then
    Standard EIF update: $(\bar{\eta}_t,\bar{\Lambda}_t) \xrightarrow{(16\text{-}18)} (\eta_t,\Lambda_t)$
else
    Marginalize: $(\bar{\eta}_t,\bar{\Lambda}_t) \xrightarrow[\mathbf{x}=\{\mathbf{x}_t,\mathbf{c}\},\mathbf{M}=\{\mathbf{m}_1\ldots\mathbf{m}_n\}]{(31\text{-}32)} (\breve{\eta}_t,\breve{\Lambda}_t)$
    while need new node do
        Augment new node: $(\breve{\eta}_t,\breve{\Lambda}_t) \xrightarrow{(14,44)} (\breve{\eta}_t,\breve{\Lambda}_t)$
    end
    Augment blank: $(\breve{\eta}_t,\breve{\Lambda}_t) \xrightarrow[\mathbf{x}=\{\mathbf{x}_t,\mathbf{c}\}]{(41,42)} (\breve{\eta}_t,\breve{\Lambda}_t)$
    Standard EIF update: $(\breve{\eta}_t,\breve{\Lambda}_t) \xrightarrow{(16\text{-}18)} (\eta_t,\Lambda_t)$
end
```

Algorithm 1: ESEIF algorithm for Vector Field SLAM

For recovering an estimate of the mean μ_t we apply Eq. (39) where the local portion is $\{\mathbf{x}_t, \mathbf{c}, \mathbf{m}_{i_0} \ldots \mathbf{m}_{i_3}\}$. In the same way we obtain an estimate of covariance Σ_{xx} by replacing η_t in (39) with unit vectors corresponding to the robot variables.

The last missing piece in using the ESEIF for Vector Field SLAM is the initialization of state and new nodes. This follows the same steps as in EKF-SLAM. The initial state is found as

$$\eta_0 = \Sigma_0^{-1}\mu_0 \qquad \Lambda_0 = \Sigma_0^{-1} \tag{43}$$

where μ_0 and Σ_0 are computed by Eqs. (12,13). Inverting Σ_0 is trivial since the matrix is diagonal.

When extrapolating a new node through Eq. (14), the mean and covariance of the new node $\mathbf{m}_{n+1}$ are computed [5]. This requires knowledge of the mean and covariance of the participating nodes $\mathbf{m}_{j_1}$ and $\mathbf{m}_{j_2}$. While we maintain an estimate of the full mean vector, the covariances of $\mathbf{m}_{j_1}$ and $\mathbf{m}_{j_2}$ have to be recovered, an operation that is expensive in the information form. We approximate the covariance $\Sigma_{m_j m_j}$ of node $\mathbf{m}_j$ by conditioning it on all map features except those contained in the node's Markov blanket $\partial\mathbf{m}_j$, i.e. those features that share information with $\mathbf{m}_j$. Since each node is the corner of at most four cells, we have at most 8 nodes in $\partial\mathbf{m}_j$. The node covariance is then computed as:

$$\Sigma_{m_j m_j} = \left(\Lambda_{m_j m_j} - \Lambda_{m_j \partial m_j}\Lambda^{-1}_{\partial m_j \partial m_j}\Lambda_{\partial m_j m_j}\right)^{-1} \tag{44}$$

and is cubic in the size of the Markov blanket. It has been noted that this approximation can result in over-confident covariances [15]. We will revisit this issue in our conclusions.

Algorithm 1 summarizes our application of the ESEIF for Vector Field SLAM. The computation time is dominated by the covariance computation (44) for augmenting new nodes and the recovery of the mean vector (39). Fortunately new nodes are added rather infrequently and the time for computing (44) is constant since $|\partial\mathbf{m}_j| \le 8$. The time for mean recovery (39) is also constant, since the size of the local portion is constant and its Markov blanket contains only nodes that share information with the four nodes of the local portion. Since each node has at most 8 neighbors, the size of the Markov blanket is constant. Thus, Vector Field SLAM using the ESEIF is an algorithm with constant time.

The memory requirements of the ESEIF are linear in the number of map nodes, due to the fact that each node shares information with at most 8 neighboring ones. Since the number of nodes is approximately proportional to the covered area of the environment, our algorithm requires memory linear in the size of the explored environment.

In the ESEIF marginalization (31,32) we chose to integrate out both robot pose $\mathbf{x}_t$ and sensor calibration $\mathbf{c}$. While it is possible to marginalize only over $\mathbf{x_t}$, the inclusion of $\mathbf{c}$ has two advantages. First, when augmenting the state with blank robot variables (41), process noise can be added to the sensor calibration through the covariance $\mathbf{R}_0$. This allows the sensor calibration to change over time or space, and ensures numerical stability. Without marginalizing over $\mathbf{c}$ any addition of process noise eventually fully populates Λ_t.

The second advantage of marginalizing out $\mathbf{c}$ is the constant size of the Markov blanket in (39). If $\mathbf{c}$ is not removed from the state, it creates shared information with all nodes and the time complexity of mean recovery becomes linear in the size of the explored area.

V. Implementation with Active Beacons

We have implemented Vector Field SLAM using ESEIF on a system using Northstar, a low-cost optical sensing system for indoor localization [16]. A beacon projects a pair of unique infrared patterns on the ceiling (see Fig. 3). An optical sensor on the robot detects these patterns and measures the direction to both spots. The sensor then reports the coordinates of both direction vectors projected onto the sensor plane.

Under ideal circumstances the reported spot coordinates change linearly with the robot position. However, infrared light reaches the sensor not only by direct line of sight but also through multiple paths by reflecting off walls and other objects, so the spot coordinates change in a non-linear way as the robot approaches an obstructed area. Fig. 4 shows the reported spot coordinate when moving the sensor along a straight line orthogonal to a wall located on the right. While the signal is quasi-linear on the left, reflections off walls distort the signal significantly until it bends over on the right. Fig. 4 also shows the piece-wise linear approximation of the signal

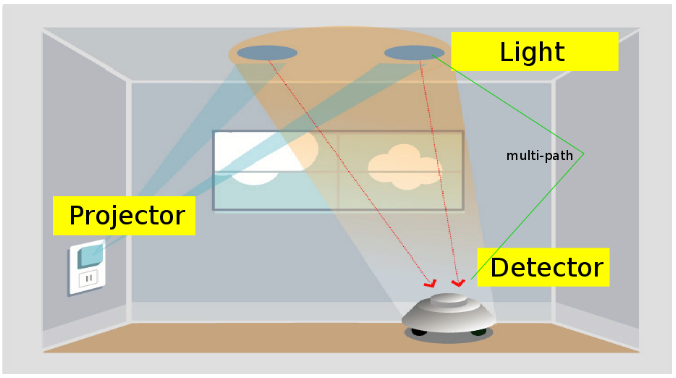

Fig. 3. The Northstar system. An optical sensor on the robot measures the bearing to two spots on the ceiling projected by an infrared beacon.

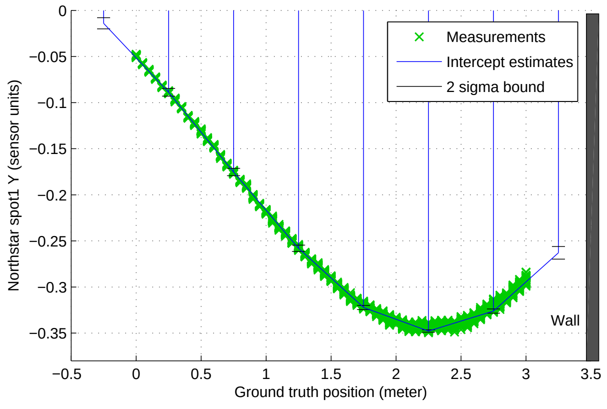

Fig. 4. Signal response of Northstar under multi-path conditions.

curve as computed by Vector Field SLAM together with 2σ intervals of the computed node covariances. Note that signal reflections can cause complex and unpredictable distortions relative to the ideal linear signal distribution.

The specific details for applying Vector Field SLAM to Northstar are as follows. The Northstar sensor provides a pair of spot coordinates, so $M = 4$ and

$$\mathbf{z}_t = (z_{x_1}, z_{y_1}, z_{x_2}, z_{y_2})^T. \quad (45)$$

The covariance $\mathbf{Q}_t$ can be derived from $\mathbf{z}_t$ along with two additional sensor outputs measuring the signal strength.

Due to tolerances in manufacturing and the mounting of the sensor on the robot, the sensor plane may not be perfectly horizontal. The result of such small angular errors is well-approximated by a coordinate offset for both spots. When rotating the sensor in place this offset becomes apparent as rotational variability. Thus, the calibration parameters are:

$$\mathbf{c} = (c_x, c_y)^T. \quad (46)$$

In the ideal case, the offset vanishes and we set $\hat{\mathbf{c}} = (0, 0)^T$.

When turning the sensor the spot coordinates change according to the rotation angle θ but in the opposite direction. The rotational component h_R of our model then becomes:

$$h_R(h_{x_1}, h_{y_1}, h_{x_2}, h_{y_2}, \theta, c_x, c_y) = \quad (47)$$
$$\begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} h_{x_1} \\ h_{y_1} \\ h_{x_2} \\ h_{y_2} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \\ c_x \\ c_y \end{pmatrix}$$

where $(h_{x_1}, h_{y_1}, h_{x_2}, h_{y_2})^T$ is the output vector of (2).

For the extrapolation of a node according to (14) we only consider node pairs which are equally spaced apart from the new node, and where the closer node is in the 8-neighborhood. The extrapolation factors in $\mathbf{A}_1$ and $\mathbf{A}_2$ are then set such that the direction between Northstar spots is copied from the closer node [5]:

$$\mathbf{A}_1 = -\frac{1}{2}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \mathbf{A}_2 = \frac{1}{2}\begin{pmatrix} 3 & 0 & 1 & 0 \\ 0 & 3 & 0 & 1 \\ 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 3 \end{pmatrix}. \quad (48)$$

This completes the implementation for Northstar.

For the motion model (8) we employ a standard odometry model [13] where at each time step the robot translates along a direction and then rotates. Jacobian $\mathbf{G}_x$ and its inverse are:

$$\mathbf{G}_x = \begin{pmatrix} 1 & 0 & -\delta_y \\ 0 & 1 & \delta_x \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{G}_x^{-1} = \begin{pmatrix} 1 & 0 & \delta_y \\ 0 & 1 & -\delta_x \\ 0 & 0 & 1 \end{pmatrix} \quad (49)$$

where $\delta_x = x_t - x_{t-1}$ and $\delta_y = y_t - y_{t-1}$.

The motion covariance $\mathbf{R}_t$ is derived from input $\mathbf{u}_t$ [13].

VI. Results

For validating Vector Field SLAM using the ESEIF we collected sensor data of a robotic vacuum cleaner equipped with Northstar. Additionally, an optical motion capture system was used to obtain ground truth data [8]. We evaluated Vector Field SLAM on 9 different runs with increasing difficulty in distortion of the Northstar signal by multi-path. For all experiments we used a cell size of 1 meter in the vector field.

Fig. 5 shows the odometry data of the robot on run 3 as computed from its wheel encoders. In this and the following figures we superimpose the ground truth trajectory by finding scale and rigid transformation that best aligns the data with the one of the motion capture system.

The result of running ESEIF-SLAM on this data is shown in Fig. 6 where small ellipses drawn every 25 cm indicate the 1 σ pose uncertainty as computed by the filter. Compare this to the result obtained by EKF-SLAM in Fig. 7. Both methods compute the path of the robot close to the ground truth positions. The ESEIF results were obtained by setting $\mathbf{R}_0 = \mathrm{diag}(5^2\mathrm{cm}^2, 5^2\mathrm{cm}^2, 0.05^2\mathrm{rad}^2)$ when computing the prior in (42). Larger diagonals in $\mathbf{R}_0$ produce more conservative estimates while smaller ones cause over-confidence. The percentage of positions falling into a Mahanalobis distance of 4.61 of the filter estimates are 91 % and 92 % for ESEIF-SLAM and EKF-SLAM respectively. This indicates that both filters compute similar estimates and that the computed Gaussians have reasonable parameters.

Fig. 8 shows the mean position errors of odometry, pose directly computed from Northstar [16], EKF-SLAM, ESEIF-

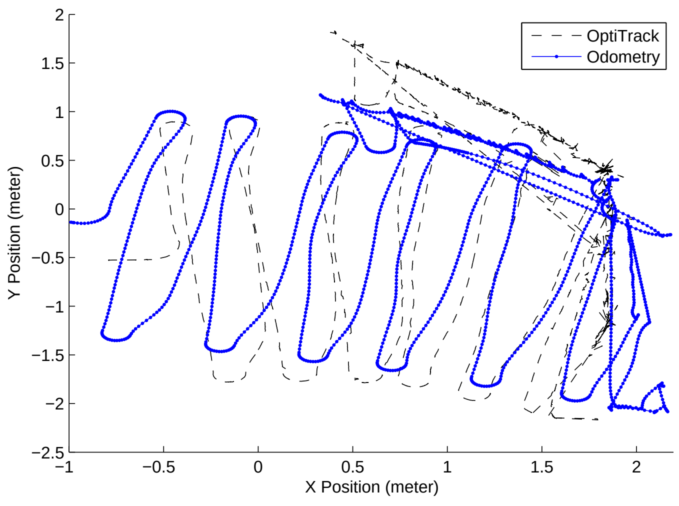

Fig. 5. Odometry information of vacuum cleaner on run 3.

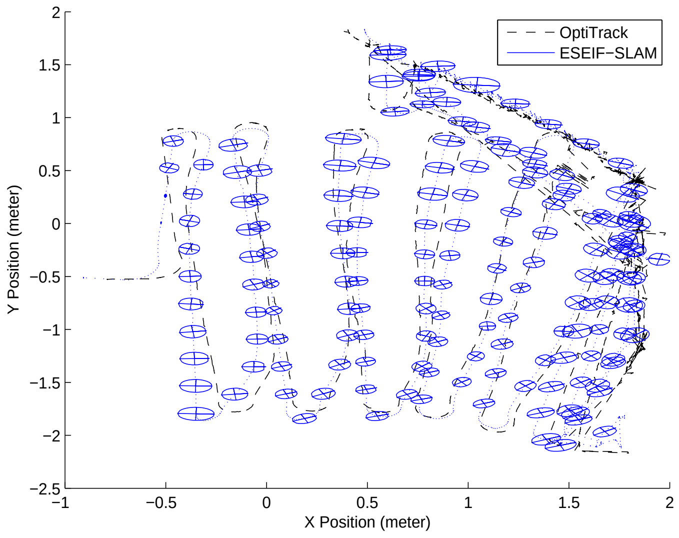

Fig. 6. Trajectory result of ESEIF-SLAM on run 3.

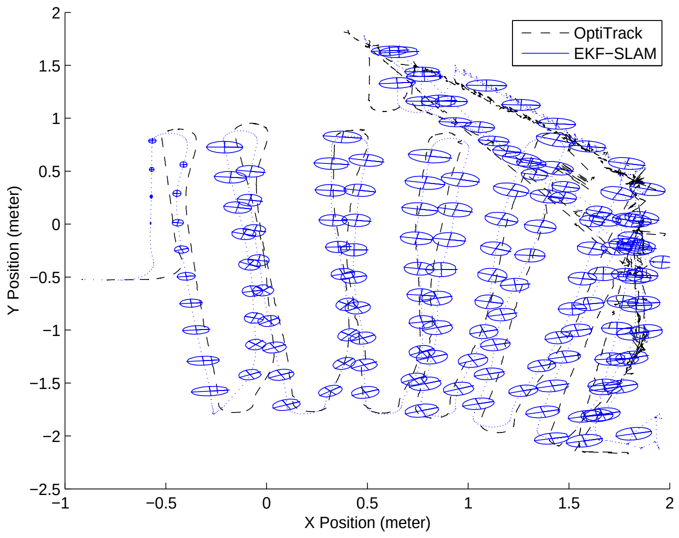

Fig. 7. Trajectory result of EKF-SLAM on run 3.

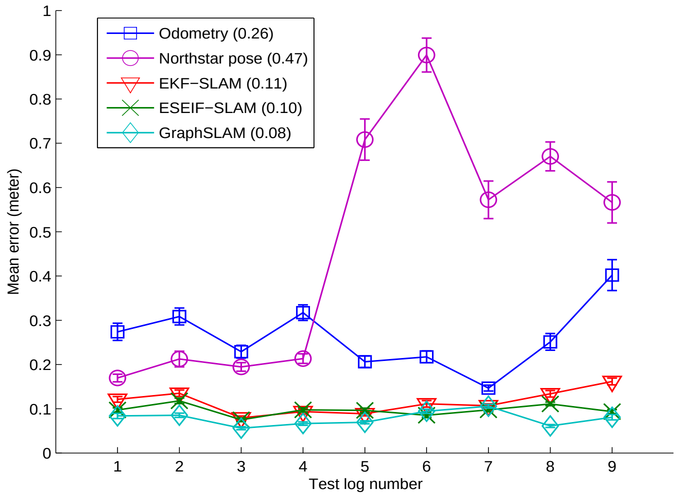

Fig. 8. Mean position errors over all 9 runs.

SLAM and an off-line GraphSLAM version [5] over all runs. Note how the error of Northstar increases from runs 1–4 to runs 5–9. The SLAM methods are able to cope with this as their mean error is not affected much. The overall mean error of ESEIF-SLAM is 10 cm which is slightly better than that of EKF-SLAM (11 cm) but worse than the full non-linear optimization of GraphSLAM (8 cm). All methods compare well to plain odometry (26 cm) or raw Northstar (47 cm).

We also implemented ESEIF-SLAM on an ARM 7 board with 64kByte memory connected to a Roomba 510 vacuum cleaner. A control program moves the robot around covering the environment in a systematic way based on the pose estimates provided by the ESEIF. We ran our system in a standard 5×4 meter test environment which is under discussion at the International Electrotechnical Commission (IEC) for evaluating the navigation capabilities of robotic floor cleaners. Fig. 9 shows the layout of this room and the trajectory of one robot run. The room is fully surrounded by walls. Solid blue (dark gray) areas mark obstacles like sofa, TV stand, and chair and table legs. The robot path starts in the upper right corner and is indicated as a white line. Areas covered by the vacuum cleaner are drawn in green (gray) while light red (light gray) marks unexplored terrain. The total run-time was 21 minutes.

One component of the vector field computed by the ESEIF in this run is shown in Fig. 10. While the sensor map contains large smooth areas, there are several non-linear regions. The big valley on the left corresponds to the sofa (largest obstacle block in Fig. 9). Furthermore, there are hills towards the front wall and the upper left corner.

We performed a total of 12 runs starting in each of the four corners of the IEC room, with the robot facing in three different directions. The position accuracy of all runs is shown in Fig. 11. The overall mean position error of the ESEIF is now 20 cm, which we attribute to the larger environment size.

For the computation of the ESEIF in these runs, a total of 42 nodes was necessary to cover the 6×5 meter environment. This leads to $3 + 2 + 42 \times 4 = 173$ variables in the ESEIF state. Thanks to the linear memory requirements, the size of the total data structure for ESEIF (including mean and robot covariance) fits into 12 kByte memory leaving room for covering larger environments and for other control structures.

The ARM 7 spends about 9 ms for a motion update and 39 ms for integrating a sensor observation. The most expensive operation is the recovery of mean and robot covariance, which involves solving a linear equation system in 21 variables through Cholesky decomposition. In summary, the total run-time of the ESEIF is about 48 ms and allows running localization with frame rates of up to 20 Hz.

VII. Conclusion

Vector Field SLAM using the ESEIF is a powerful tool for learning the distribution of time-stationary signals over the environment. By limiting the number of active features to the four nodes of the cell enclosing the robot, we obtain a constant-time SLAM algorithm with linear memory requirements. The cost of this achievement is a tradeoff in the

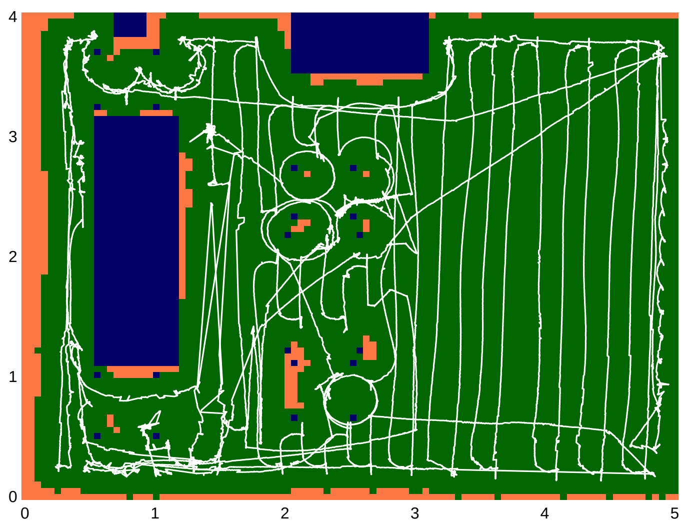

Fig. 9. Navigation run of Roomba 510 using the ESEIF in IEC room.

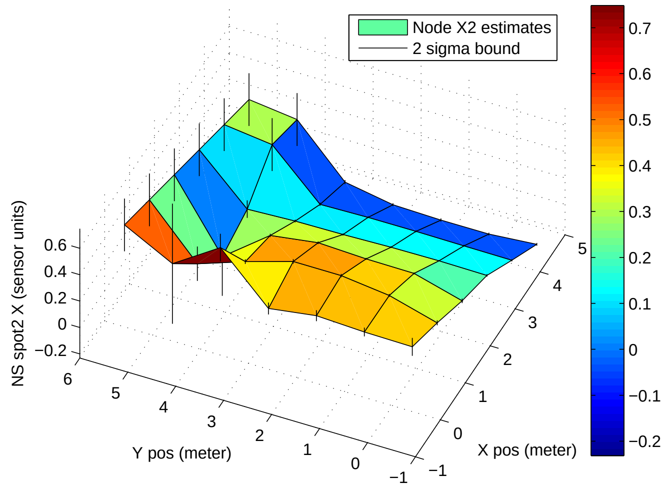

Fig. 10. Spot 2 X component estimated by ESEIF-SLAM in the IEC room.

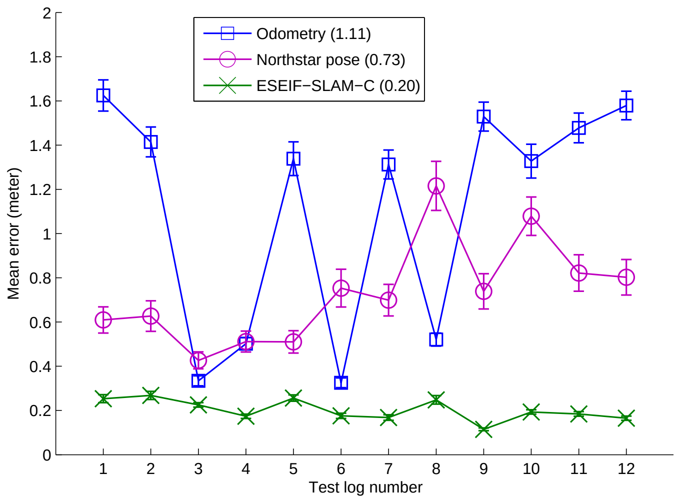

Fig. 11. Mean position errors over all 12 runs in the IEC room.

consistency of the filter. In order to arrive at our formulation, a number of approximations were made:

- Our procedure for relocating the robot requires the tuning of a prior and may result in over-confident estimates.
- The extrapolation of new nodes approximates covariances using Markov blankets which can result in over-confident covariances [15]. This can be compensated to some extent by using a larger preset covariance $\mathbf{S}$ in (14).
- The mean recovery computes only the values of robot pose, calibration and the nodes of the current cell. This may be insufficient in particular when closing loops.
- Like in other on-line SLAM methods, the errors introduced by evaluating motion and sensor Jacobians at the latest state estimate may cause inconsistent results [7].

Our experiments show that ESEIF-SLAM is capable of keeping a robotic vacuum cleaner localized in a home environment using low-cost sensors on embedded hardware. This opens the door for location-aware robotic consumer products such as autonomous and systematic floor cleaners.

REFERENCES

[1] B. Ferris, D. Fox, and N. Lawrence. WiFi-SLAM using Gaussian process latent variable models. In *Int. Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[2] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24:381395, 1981.

[3] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[4] J. Graefenstein and M.E. Bouzouraa. Robust method for outdoor localization of a mobile robot using received signal strength in low power wireless networks. In *Int. Conference on Robotics and Automation (ICRA)*, 2008.

[5] J.-S. Gutmann, G. Brisson, E. Eade, P. Fong, and M. Munich. Vector Field SLAM. In *Int. Conference on Robotics and Automation (ICRA)*, Anchorage, 2010.

[6] A. Haeberlen, E. Flannery, A.M. Ladd, A. Rudys, D.S. Wallach, and L.E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proc. Int. Conference on Mobile Computing and Networking (MOBICOM)*, 2004.

[7] G.P. Huang, A.I. Mourikis, and S.I. Roumeliotis. Analysis and improvement of the consistency of Extended Kalman Filter based SLAM. In *Int. Conference on Robotics and Automation (ICRA)*, 2008.

[8] NaturalPoint Inc. Optitrack camera system, www.optitrack.com, 2009.

[9] Woo Yeon Jeong and Kyoung Mu Lee. CV-SLAM: A new ceiling vision-based SLAM technique. In *Int. Conference on Intelligent Robots and Systems (IROS)*, 2005.

[10] K. Konolige. Large-scale map-making. In *National Conference on Artificial Intelligence (AAAI)*, 2004.

[11] K. Konolige, J. Augenbraun, N. Donaldson, C. Fiebig, and P. Shah. A low-cost laser distance sensor. In *Int. Conference on Robotics and Automation (ICRA)*, 2008.

[12] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

[13] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[14] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Int. Journal of Robotics Research*, 23(7-8), 2004.

[15] Matthew R. Walter, Ryan M. Eustice, and John J. Leonard. Exactly sparse extended information filters for feature-based SLAM. *International Journal of Robotics Research*, 26(4):335–359, 2007.

[16] Y. Yamamoto, P. Prijanian, J. Brown, M. Munich, E. Di Bernardo, L. Goncalves, J. Ostrowski, and N. Karlsson. Optical sensing for robot perception and localization. In *Proc. Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2005.