2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Vector Field SLAM

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

***Abstract*— Localization in unknown environments using low-cost sensors remains a challenge. This paper presents a new localization approach that learns the spatial variation of an observed continuous signal. We model the signal as a piece-wise linear function and estimate its parameters using a simultaneous localization and mapping (SLAM) approach. We apply our framework to a sensor measuring bearing to active beacons where measurements are systematically distorted due to occlusion and signal reflections of walls and other objects present in the environment. Experimental results from running GraphSLAM and EKF-SLAM on manually collected sensor measurements as well as on data recorded on a vacuum-cleaner robot validate our model.**

I. Introduction

Localization is one of the most important components in a robot system for ensuring reliable navigation, prompting the development of a manifold of approaches using a variety of sensors [1]. Localization usually requires a map of the environment to be available, which can be hand-crafted or obtained through the robot's sensors [9].

In this research, we are interested in localizing a mobile robot without the need of an *a priori* map and requiring only minimal modifications to the environment. The solution should be low cost and provide pose estimates in real-time. Such a system would be useful, for example, in floor cleaning where systematic area coverage is desired.

In our approach a set of signals is emitted into the environment and measured by a sensor on the robot. We impose the following requirements on the signals:

1) Each signal source can be uniquely identified.
2) Signals are spatially continuous and constant over time.
3) At any fixed position, how the signal varies with orientation can be fully described by sensor orientation and internal sensor parameters.

Examples of such signals are the signal strengths to WiFi stations or cellular networks, or bearings to unique beacons.

Req. 1 eliminates the data association problem. Signals in WiFi and other networks contain a unique ID as part of their data packet protocol. Active beacons like the ones used in our experiments are identified by their modulation frequency.

Req. 2 ensures that the emitted signals constitute a vector field that varies over space but is stationary in time. Note that there are no restrictions other than continuity: the same measurement vector might be observed at several different locations in the environment (the measurement model need not be injective). Such scenarios occur, for example, when signals emitted by active beacons bounce off walls and other objects. As a result, the observed bearing can be very different than the actual one (see Fig. 1). A given vector of such observations can match to several distinct positions.

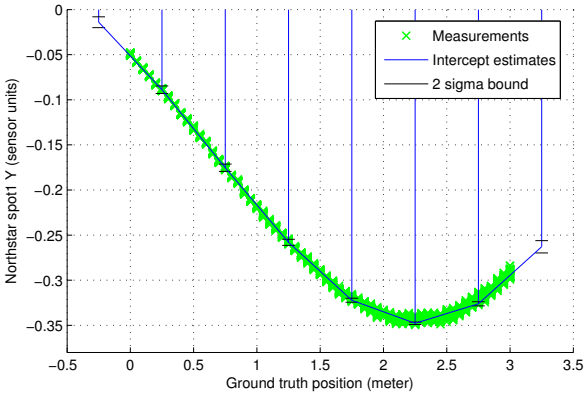


Fig. 1. Sensor response of an active beacon signal as the sensor approaches a wall on the right. Reflections from the wall cause non-linear distortion. A piece-wise linear approximation estimated by our filter is superimposed.

Finally, Req. 3 implies that given the signal values at one sensor position and orientation, we can predict the values for other orientations at the same position, independent of the position itself. For example, a WiFi receiver might show changes in signal strength when rotating, caused solely by the directional sensitivity of the antenna. Similarly, a sensor measuring bearing and elevation to beacons can show variations due to calibration errors of the sensor's vertical axis. We call such signal changes *rotational variability*, and assume such variability is not a function of position.

In order to localize the robot from an initial start position, we formulate the problem as one of simultaneous localization and mapping (SLAM). The map is an observation function describing the observed vector field over space. We model this vector field using a piece-wise linear function arising from estimates of the vector signals at pre-defined ground positions called *nodes* with the sensor facing a fixed orientation. The full SLAM state then contains the robot path, rotational variability and the signal values at all nodes.

Signal values at arbitrary locations are predicted using bilinear interpolation. This allows us to apply standard solutions such as GraphSLAM and EKF-SLAM to compute estimates of the SLAM state. With respect to navigation, we are mainly interested in the localization of the robot, i.e. the robot path. The learned rotational variability and node estimates are of secondary interest but are still useful for sensor evaluation, environment analysis or re-localization.

Through experiments we show that our approach accurately localizes a robot in an environment equipped with active beacons. In order to demonstrate the advantage of our model, we compare the trajectories computed by Vector Field SLAM with those of an EKF that does not learn a signal map.

The remainder of this paper is organized as follows. After discussing related work in the next section we present our SLAM approach for learning a continuous vector field in Section III. An implementation on a sensor system using active beacons is described in Section IV followed by experimental results and our conclusions.

II. Related Work

Localization from signals containing unique IDs has been addressed long ago in navigation systems using active beacons [1]. Most of these systems measure range or bearing to known landmarks and compute a pose by trilateration or triangulation. These systems start to fail when signals are observed not only through the direct path of flight but also through reflections off walls or other objects. Our approach provides a direct extension addressing this problem.

An early formulation of the SLAM problem utilizes landmarks as the map features [8]. The state contains the robot pose and the positions of all landmarks. Our method does not estimate the positions of landmarks but the signal values at fixed ground positions. Our approach shares, however, the statistical techniques for computing state estimates.

Localization using the signal strengths of WiFi has become a focus in several research domains. If the positions of base stations are known, localization can be achieved with remarkable accuracy [5]. The main challenge is obtaining a sensor map of ground positions to signal strengths for localization. There are practical solutions for topological localization in a large office environment [6], although these require a training phase for obtaining the signal map. An approach using Gaussian Process Latent Variable Models allows building of topologically correct maps from collected signal strength data [3]. Our approach estimates similar signal maps but learns a piece-wise linear approximation of the vector field directly through SLAM.

III. Vector Field SLAM

The basic idea of our approach for localizing a mobile robot moving through a field of vector signals is to learn the signal distribution over space and at the same time track the pose of the robot. This is known as SLAM.

A. Simultaneous Localization and Mapping

In SLAM, a robot moves through a time series of poses $\mathbf{x}_0 \dots \mathbf{x}_T$, $\mathbf{x}_t \in \mathrm{SE}(2)$ in an environment containing N map features $\mathbf{m}_1 \dots \mathbf{m}_N$, $\mathbf{m}_i \in \mathbb{R}^M$. Without loss of generality, $\mathbf{x}_0 = (0,0,0)^T$. At each time step $t = 1 \dots T$ 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:

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

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

Furthermore, a sensor model defined by a function h predicts an observation given current robot pose and features:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{m}_1 \dots \mathbf{m}_N) + \mathbf{e}_z \tag{2}$$

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

The state of the online SLAM problem [9] is:

$$\mathbf{y}_t = (\mathbf{x}_t, \mathbf{m}_1 \dots \mathbf{m}_N)^T. \tag{3}$$

A method that recursively estimates this state is the Extended Kalman Filter (EKF). Starting from an initial mean μ_0 and covariance Σ_0 the state is updated on motion according to:

$$\bar{\mu}_t = f(\mu_{t-1}, \mathbf{u}_t) \tag{4}$$
$$\bar{\Sigma}_t = \mathbf{F}_y \Sigma_{t-1} \mathbf{F}_y^T + \mathbf{R}_t \tag{5}$$

where f extends the motion model g over all state variables and $\mathbf{F}_y$ is its Jacobian with respect to state $\mathbf{y}_t$:

$$f(\mathbf{y}_{t-1}, \mathbf{u}_t) = (g(\mathbf{x}_{t-1}, \mathbf{u}_t), \mathbf{m}_1 \dots \mathbf{m}_N)^T \tag{6}$$
$$\mathbf{F}_y = \frac{\partial f}{\partial \mathbf{y}}(\mu_{t-1}, \mathbf{u}_t). \tag{7}$$

Given a sensor observation $\mathbf{z}_t$, the state is updated as:

$$\mu_t = \bar{\mu}_t + \mathbf{K}(\mathbf{z}_t - h(\bar{\mu}_t)) \tag{8}$$
$$\Sigma_t = \bar{\Sigma}_t - \mathbf{K}\mathbf{H}_y \bar{\Sigma}_t \tag{9}$$

where $\mathbf{H}_y$ is the Jacobian of h and $\mathbf{K}$ the Kalman gain:

$$\mathbf{H}_y = \frac{\partial h}{\partial \mathbf{y}}(\bar{\mu}_t) \tag{10}$$
$$\mathbf{K} = \bar{\Sigma}_t \mathbf{H}_y^T (\mathbf{H}_y \bar{\Sigma}_t \mathbf{H}_y^T + \mathbf{Q}_t)^{-1}. \tag{11}$$

In many implementations of EKF-SLAM the initial state contains no or only a minimal set of map features. The state is augmented with new features as they appear. Initialization of new features can add significant complexity such as performing non-linear optimization on a short sequence of measurements or reformulating the observation model in a transformed space [2]. The properties and limitations of EKF-SLAM are well understood [9].

In contrast, the state of the full SLAM problem [9] is

$$\mathbf{Y} = (\mathbf{x}_1 \dots \mathbf{x}_T, \mathbf{m}_1 \dots \mathbf{m}_N)^T. \tag{12}$$

GraphSLAM is a non-linear optimization method that computes $\mathbf{Y}$ as the solution minimizing an objective function:

$$J = \sum_{t=1}^{T} (\mathbf{x}_t - g(\mathbf{x}_{t-1}, \mathbf{u}_t))^T \mathbf{R}_t^{-1} (\mathbf{x}_t - g(\mathbf{x}_{t-1}, \mathbf{u}_t)) + \sum_{t=1}^{T} (\mathbf{z}_t - h(\mathbf{y}_t))^T \mathbf{Q}_t^{-1} (\mathbf{z}_t - h(\mathbf{y}_t)). \tag{13}$$

GraphSLAM is a batch processing instance of *Bundle Adjustment* [10] and requires all data be present for computation. A possible implementation uses the following steps:

1) Provide an initial state estimate for (12).
2) Linearize (13) using the Jacobians in (7) and (10) of the current estimate. This results in a quadratic function.
3) Solve the linear equation system that minimizes the quadratic function through the Conjugate Gradient method. This provides an improved estimate of (12).
4) Repeat steps 2 and 3 until the solution converges.

Assuming a good initial estimate is provided, GraphSLAM computes the best possible estimate, i.e. the estimate with the lowest variance that any unbiased estimator can have [10], given the model induced by (1) and (2). Although the method does not output estimates online, it has many successful applications [9], [10]. In this work, we use GraphSLAM as a baseline algorithm for validating our SLAM model.

B. SLAM with Sensor Calibration

In practical applications, the measurement $\mathbf{z}_t$ is often affected by the internal calibration of the sensor. Using SLAM this calibration can be estimated along with the robot path and map features by including a vector $\mathbf{c}$ of calibration parameters in the observation model (2) and state vector (3) respectively (12):

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{c}, \mathbf{m}_1 \dots \mathbf{m}_N) + \mathbf{e}_z \quad (14)$$

$$\mathbf{y}_t = (\mathbf{x}_t, \mathbf{c}, \mathbf{m}_1 \dots \mathbf{m}_N)^T \quad (15)$$

$$\mathbf{Y} = (\mathbf{x}_1 \dots \mathbf{x}_T, \mathbf{c}, \mathbf{m}_1 \dots \mathbf{m}_N)^T. \quad (16)$$

EKF-SLAM and GraphSLAM then compute estimates for the calibration parameters using the same equations presented above. We will make use of this technique to learn the vector field and the rotational variability of a sensor encoded as calibration parameters in the next section.

C. Application to Vector Fields

We assume there is a time-invariant vector field of dimension M defined over the environment satisfying the constraints in Reqs. 1-3. Our goal is to learn the observation model or vector field

$$\text{VF} : \text{SE}(2) \rightarrow \mathbb{R}^M \quad (17)$$

that maps a ground pose to a vector of signal values.

Since signals are independent of sensor orientation (Req. 3) we decompose the space of poses $\text{SE}(2)$ 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 $\mathbf{b}_i = (b_{i,x}, b_{i,y})^T$, $i = 1 \dots N$ 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 placing the sensor at $\mathbf{b}_i$ and pointing at a pre-defined direction θ_0.

For an arbitrary sensor position with orientation θ_0, the signal values are computed by bilinear interpolation from the four nodes of the cell containing the sensor position. Let $\mathbf{x}_t = (x, y, \theta)^T$ be the sensor pose and $\mathbf{b}_{i_0} \dots \mathbf{b}_{i_3}$ the cell

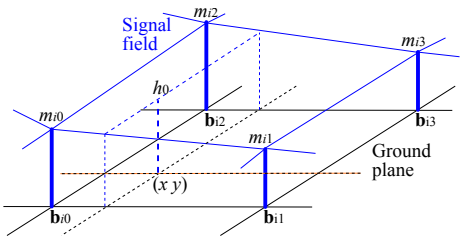


Fig. 2. Bilinear interpolation from cell nodes

nodes enclosing the sensor as shown in Fig. 2. The signal values at (x, y) with orientation θ_0 are then computed as:

$$h_0(x, y, \mathbf{m}_1 \dots \mathbf{m}_N) = \sum_{j=0}^{3} w_j \mathbf{m}_{i_j} \quad (18)$$

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

$$w_0 = \frac{(b_{i_1,x} - x)(b_{i_2,y} - y)}{(b_{i_1,x} - b_{i_0,x})(b_{i_2,y} - b_{i_0,y})} \quad (19)$$

$$w_1 = \frac{(x - b_{i_0,x})(b_{i_2,y} - y)}{(b_{i_1,x} - b_{i_0,x})(b_{i_2,y} - b_{i_0,y})} \quad (20)$$

$$w_2 = \frac{(b_{i_1,x} - x)(y - b_{i_0,y})}{(b_{i_1,x} - b_{i_0,x})(b_{i_2,y} - b_{i_0,y})} \quad (21)$$

$$w_3 = \frac{(x - b_{i_0,x})(y - b_{i_0,y})}{(b_{i_1,x} - b_{i_0,x})(b_{i_2,y} - b_{i_0,y})}. \quad (22)$$

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

$$h(\mathbf{x}_t, \mathbf{c}, \mathbf{m}_1 \dots \mathbf{m}_N) = h_R(h_0(x, y, \mathbf{m}_1 \dots \mathbf{m}_N), \theta, \mathbf{c}). \quad (23)$$

Here h_R is a continuous function that transforms the interpolated signal values obtained through (18) by the sensor orientation and calibration. This is usually a rotation by orientation 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 IV.

Eq. (23) shows our factorization of the sensor map into two components h_0 and h_R following the requirements set forth in the introduction. Component h_0 depends only on position (Req. 2) whereas component h_R depends only on orientation, sensor calibration and h_0 (Req. 3).

Note that Eq. (18) implies a model selection. Given the current estimate of the sensor position, the nodes used in the bilinear interpolation are selected. This can result in the wrong model if the position estimate is far from the actual position. We revisit this issue in our conclusions.

D. Node Initialization

The final missing piece for applying a SLAM method to our model is how to initialize the nodes. In general this depends on the type of signals and the sensor used. We envision, however, that the following procedure is applicable to most systems.

We assume the robot collects a short sequence of motion and sensor measurements $\mathbf{u}_1, \mathbf{z}_1 \dots \mathbf{u}_{t_0}, \mathbf{z}_{t_0}$ and has a rough idea $\hat{\mathbf{c}}$ of its rotational variability. By iteration of the motion model (1) from the initial pose $\mathbf{x}_0$ we obtain a path $\hat{\mathbf{x}}_1 \dots \hat{\mathbf{x}}_{t_0}$ with each $\hat{\mathbf{x}}_t = (\hat{x}_t, \hat{y}_t, \hat{\theta}_t)^T$. For all sensor measurements $\mathbf{z}_t$ we compute a point $\mathbf{h}_t$ in the vector field using the inverse of the rotational component:

$$\mathbf{h}_t = h_R^{-1}(\mathbf{z}_t, \hat{\theta}_t, \hat{\mathbf{c}}) \quad (24)$$

where h_R^{-1} is the inverse function of h_R in its first argument. This provides us with a set of samples $(\hat{x}_t, \hat{y}_t, \mathbf{h}_t)$ of the vector field. We then fit a linear model to this sample set defined by a base vector $\mathbf{h}_0$ and scale matrix $\mathbf{A}_0$:

$$\mathbf{h}_t = \mathbf{h}_0 + \mathbf{A}_0 \begin{pmatrix} \hat{x}_t \\ \hat{y}_t \end{pmatrix} + \mathbf{e}_0 \quad (25)$$

where $\mathbf{e}_0$ is a zero mean error with unknown covariance. Elements of $\mathbf{h}_0$ and $\mathbf{A}_0$ can be found by linear optimization. For data containing outliers, methods such as RANSAC [4] allow robust estimation of the linear model.

Equipped with the parameters of the linear model we can now initialize any node i in the state vector by applying (25) to the node position $\mathbf{b}_i$:

$$\mathbf{m}_i = \mathbf{h}_0 + \mathbf{A}_0\mathbf{b}_i + \mathbf{e}_0. \quad (26)$$

For EKF-SLAM we set the initial state to contain only the four nodes of the cell in which the robot is located:

$$\mathbf{y}_0 = (\mathbf{x}_0, \mathbf{c}, \mathbf{m}_1 \dots \mathbf{m}_4)^T. \quad (27)$$

Initial mean and covariance are then found as:

$$\mu_0 = (0, 0, 0, \hat{\mathbf{c}}, \hat{\mathbf{m}}_1 \dots \hat{\mathbf{m}}_4)^T \quad (28)$$
$$\Sigma_0 = \mathrm{diag}(0, 0, 0, \infty \dots \infty) \quad (29)$$

where $\hat{\mathbf{m}}_i = \mathbf{h}_0 + \mathbf{A}_0\mathbf{b}_i$ for $i = 1 \dots 4$. Although a tighter initial covariance can be computed from the fitting error in (25), in practice Eq. (29) is often sufficient since the node covariances decrease quickly within a few measurements.

As long as the robot stays in a cell, the EKF reduces the uncertainties in rotational variability and node values on measurements through (8, 9) while generally increasing the pose uncertainty on motion through (4, 5). When moving into another cell, new nodes may have to be added to the state. If we assume the vector field changes slowly over space we can augment the state vector with new nodes whose values are extrapolated from nodes already contained in the state.

Let $\mathbf{y}_t = (\mathbf{x}_t, \mathbf{c}, \mathbf{m_1} \dots \mathbf{m}_n)$ 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 as:

$$\mathbf{m}_{n+1} = \mathbf{A}_1\mathbf{m}_{j_1} + \mathbf{A}_2\mathbf{m}_{j_2} + \mathbf{e}_1 \quad (30)$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ are matrices containing 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. For example, if the three nodes are equally spaced apart, i.e. the distances between $\mathbf{b}_{j_1}$ and $\mathbf{b}_{j_2}$ and between $\mathbf{b}_{j_2}$ and $\mathbf{b}_{n+1}$ are equal, then an uncorrelated choice of extrapolation factors is $\mathbf{A}_1 = -I$ and $\mathbf{A}_2 = 2I$ where I is the identity matrix. Another example of extrapolation factors is shown in Fig. 4 and described in our implementation in Section IV.

Mean and covariance of the EKF are augmented as:

$$\mu_t \leftarrow \begin{pmatrix} \mu_t \\ \mathbf{A}_1\hat{\mathbf{m}}_{j_1} + \mathbf{A}_2\hat{\mathbf{m}}_{j_2} \end{pmatrix} \quad (31)$$

$$\Sigma_t \leftarrow \begin{pmatrix} \Sigma_t & \Sigma_t\mathbf{A}_{12}^T \\ \mathbf{A}_{12}\Sigma_t & \mathbf{A}_{12}\Sigma_t\mathbf{A}_{12}^T + \mathbf{S} \end{pmatrix} \quad (32)$$

where $\mathbf{A}_{12} = (0 \dots 0, \mathbf{A}_1, 0 \dots 0, \mathbf{A}_2, 0 \dots 0)$ with elements of $\mathbf{A}_1$ and $\mathbf{A}_2$ at inidices corresponding to nodes j_1 and j_2.

As long as the robot never jumps over a complete cell, we can always find a pair (j_1, j_2) for each new node in a cell. However, there might be several choices for extrapolation sources, leaving room for strategies such as choosing the node pair with the smallest resulting uncertainty.

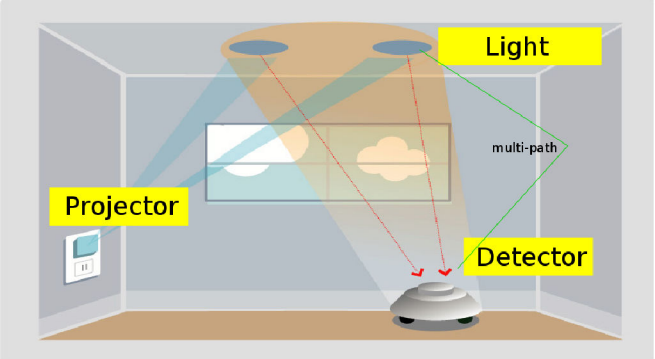


Fig. 3. The Northstar system. An optical sensor on the robot measures the bearing to two spots on the ceiling projected by an infrared beacon.

The initialization for GraphSLAM uses the same technique employed in the initialization of EKF-SLAM. The initial path is obtained by iteration of the motion model (1) from the initial pose $\mathbf{x}_0$. Using the linear model given in (26) all nodes are initialized. The full initial state becomes:

$$\mathbf{Y}^0 = (\hat{\mathbf{x}}_1 \dots \hat{\mathbf{x}}_T, \hat{\mathbf{c}}, \hat{\mathbf{m}}_1 \dots \hat{\mathbf{m}}_N)^T. \quad (33)$$

In our experiments this was sufficient to ensure convergence.

IV. Implementation with Active Beacons

We have implemented the method of Vector Field SLAM on a system using Northstar, a low-cost optical sensing system for indoor localization [11]. A beacon projects a pair of unique infrared patterns on the ceiling (see Fig. 3). The beacon can be placed relatively freely in the room and adjusted such that it points towards the ceiling. An optical sensor on the robot measures the direction to both spots on the ceiling. 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. 1 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. 1 also shows the piece-wise linear approximation of the signal curve as computed by EKF-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

$$\mathbf{z}_t = (z_{x_1}, z_{y_1}, z_{x_2}, z_{y_2})^T. \quad (34)$$

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

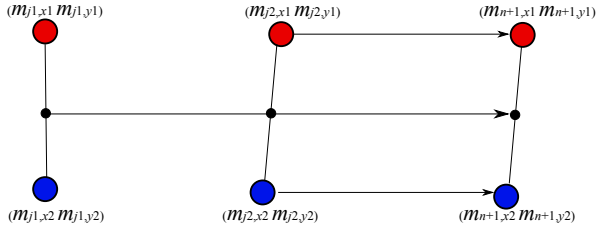


Fig. 4. Extrapolation of new node preserving direction between spots.

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:

$$\mathbf{c} \;=\; (c_x, c_y)^T. \tag{35}$$

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:

$$h_R(h_{x_1}, h_{y_1}, h_{x_2}, h_{y_2}, \theta, c_x, c_y) \;=\; \tag{36}$$
$$\begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} h_{x_1} \\ h_{y_1} \\ h_{x_2} \\ h_{y_2} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \\ c_x \\ c_y \end{pmatrix}$$

where $(h_{x_1}, h_{y_1}, h_{x_2}, h_{y_2})^T$ is the output vector of (18).

For the extrapolation of a node according to (30) we consider only node pairs that are equally spaced away 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 (see Fig. 4):

$$\mathbf{A}_1 = -\frac{1}{2}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \mathbf{A}_2 = \frac{1}{2}\begin{pmatrix} 3 & 0 & 1 & 0 \\ 0 & 3 & 0 & 1 \\ 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 3 \end{pmatrix}. \tag{37}$$

This completes the implementation for Northstar.

As for the motion model (1), we employ an odometry motion model [9] where at each time step the robot first translates along a direction and then rotates. The motion covariance $\mathbf{R}_t$ is derived from the motion input $\mathbf{u}_t$.

V. Results

We collected data from Northstar by mounting the sensor on a rail and moving it to controlled poses spaced 50 cm apart and with 8 different directions (every $45°$) in a room surrounded by walls on the left, right and far side. Fig. 5(a) shows the 8×7 grid positions where data were collected. The sensor readings of one spot, corrected by the ground truth orientation of the sensor, are displayed in Fig. 6 where each color (gray scale) corresponds to one of the 8 directions.

This illustrates two aspects of the signal distribution. First, the rotational variability of the sensor is manifested as small circles that are approximately equal in size throughout the room. The mean radius of these circles implies an angular error in the sensor plane of about $0.7°$. Furthermore, multi-path effects disturb the signal on the left, right and far side. While left and right sides show some level of signal compression, the far side and the upper left and right corners are severely distorted, caused by reflections off the walls.

We have simulated a path through the rail grid that starts in the lower left corner, moves along the rows and changes between rows on the left and right side. This results in straight forward motions along rows and two $90°$ turns with intermediate forward motions on the sides. By adding noise to the motion estimates, an odometry path is obtained as shown in Fig. 5(b). After moving up and down the rail grid about 10 times, the error in orientation has increased to $90°$.

Using this odometry path we run EKF-SLAM by choosing a random Northstar measurement from each visited ground truth pose. As soon as 5 Northstar measurements are collected, RANSAC estimates the linear model in (25) from which the initial four nodes of the vector field with a cell size of 1 m are initialized. The remaining measurements gated at 3σ are then fed into the EKF. The computed vector field for one spot is shown in Fig. 7 together with the Northstar measurements and 2σ bounds of the computed covariance. Comparing the curves of the spot coordinates to the actual sensor map in Fig. 6 shows how the estimated vector field approximates the compression from multi-path.

Fig. 5(c) shows the computed robot trajectory. Vector Field SLAM keeps the robot localized when moving up and down the rails. The mean position error in this experiment is 6 cm.

We have also applied Vector Field SLAM on a vacuum cleaner robot equipped with a Northstar sensor. Ground truth trajectories are obtained from *OptiTrack*, an optical motion capture system [7]. We evaluate our approach on 9 different runs with increasing degrees of distortion of the Northstar signal from multi-path. Fig. 8 shows the odometry data of the robot on run 4 as computed from its wheel encoders. In this and the following figures we superimpose the ground truth trajectory by finding the rigid transformation and scale that best aligns the data with the output of the motion capture system. Fig. 9 visualizes the raw sensor data as pose directly computed from the Northstar measurements [11]. Note how the signal becomes distorted on the right and far side.

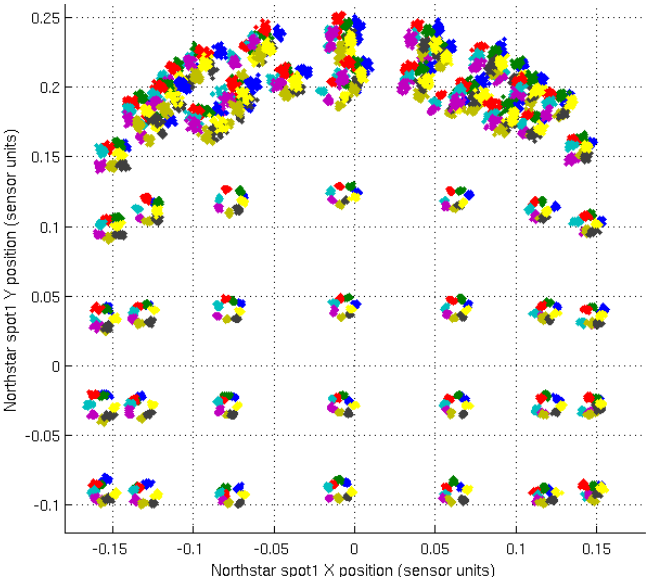
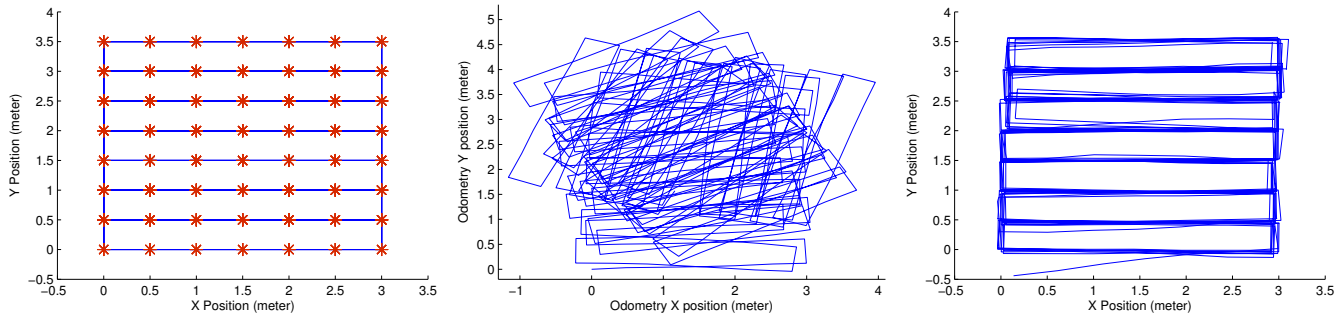


Fig. 6. Northstar data of one spot at rail positions with 8 orientations.



(a) (b) (c)

Fig. 5. Experiment with data collected on a grid of rail positions: (a) grid positions and simulated path, (b) odometry information, (c) result of EKF-SLAM.

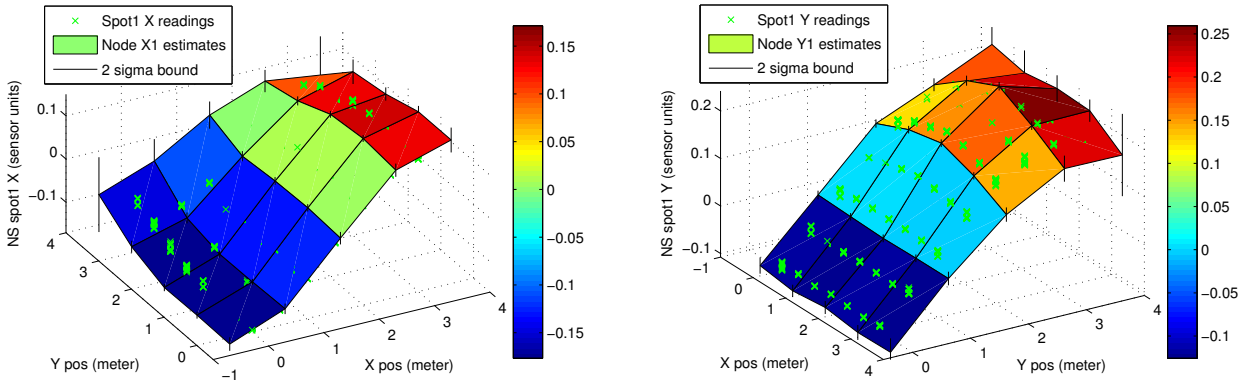


Fig. 7. Result of the vector field of x and y coordinates of one spot as learned by EKF-SLAM. The curves for the other spot are similar.

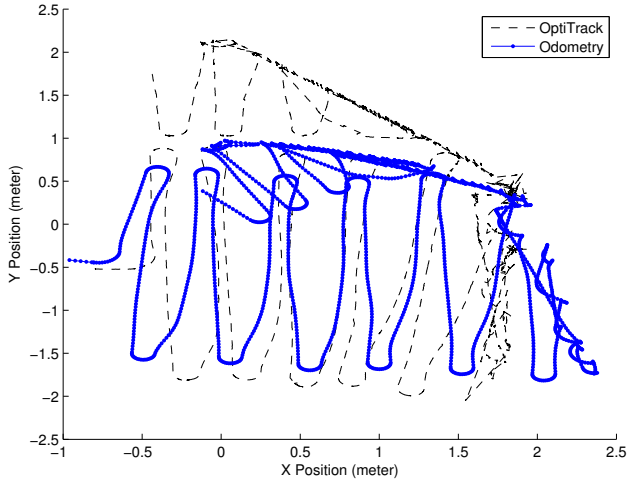


Fig. 8. Odometry information of vacuum cleaner on run 4.

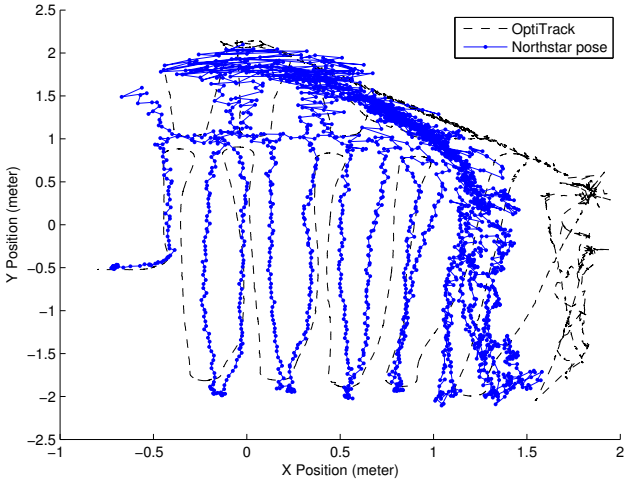


Fig. 9. Northstar pose information of vacuum cleaner on run 4.

For comparing our approach to a basic geometric method we implemented an EKF that does not learn a signal map. Besides robot pose and rotational variability the state of this EKF contains the scale between ground position and sensor response, and the distance between the two spots. This representation is equivalent to tracking both spot positions. Fig. 10 shows the trajectory result of this filter. While the method provides reasonable estimates for the left and center part of the environment, it fails when moving to the right. The large distortions caused by multi-path effects require a spatial modeling of the sensor signals.

The result of running EKF-SLAM using the same parameters as in the rail grid data is shown in Fig. 11 while Fig. 12 shows the result of GraphSLAM with 10 iterations of the algorithm outlined in Section III. Both methods are able to estimate the robot path close to the ground truth positions.

Finally Fig. 13 shows the mean localization errors 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. GraphSLAM shows the best performance with an overall error of 9 cm. EKF-SLAM is slightly less accurate (11 cm) which we believe is due to the initial error in rotational variability and the linearization errors of motion and observation models. Both methods are significantly better than the filter without signal map (26 cm), odometry (29 cm) or raw Northstar (47 cm).

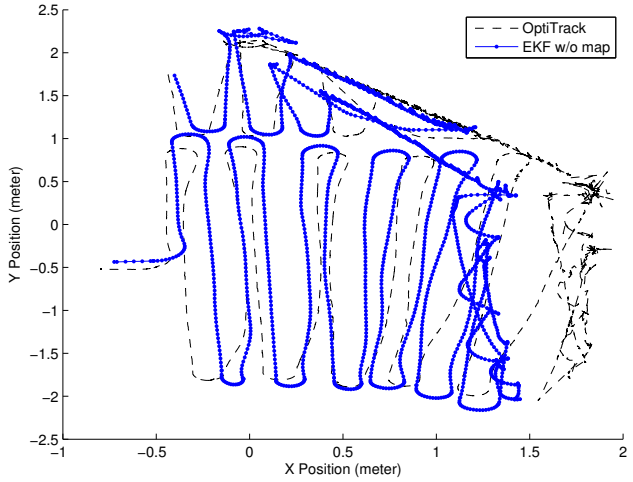


Fig. 10. Trajectory result of an EKF without a signal map on run 4.

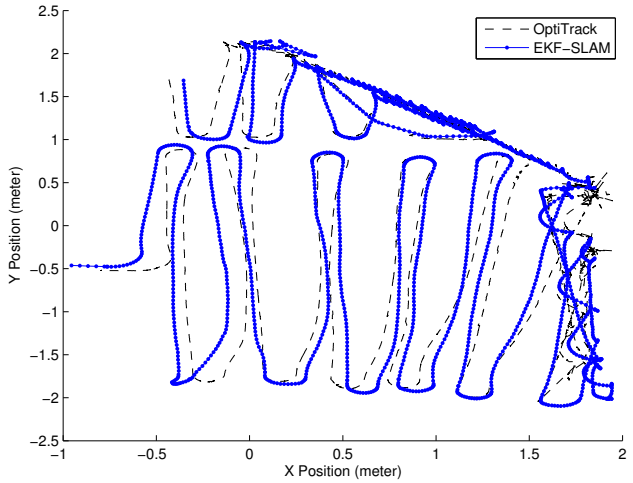


Fig. 11. Trajectory result of EKF-SLAM on run 4.

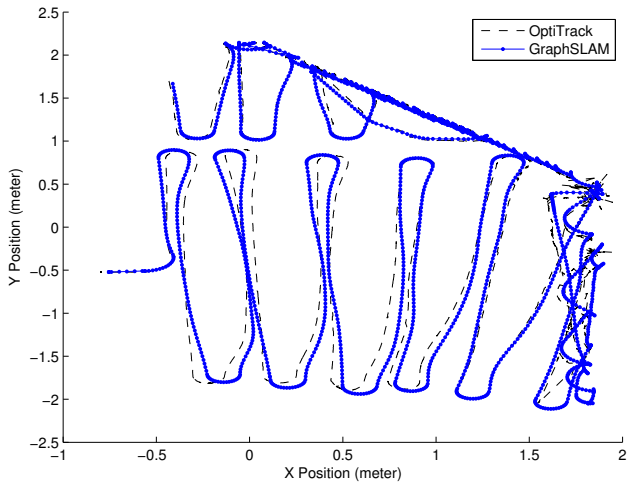


Fig. 12. Trajectory result of GraphSLAM on run 4.

All vector fields in our experiments can be represented by no more than 30 nodes. This results in a state vector for EKF-SLAM of $3 + 2 + 4 \cdot 30 = 125$ variables which allows computation of estimates in real-time on a low-end PC. For GraphSLAM the number of poses, which is more relevant to computational complexity, ranges from about 1000 to 2500 in our experiments. The non-linear optimization then takes a few minutes per run in a Matlab environment.

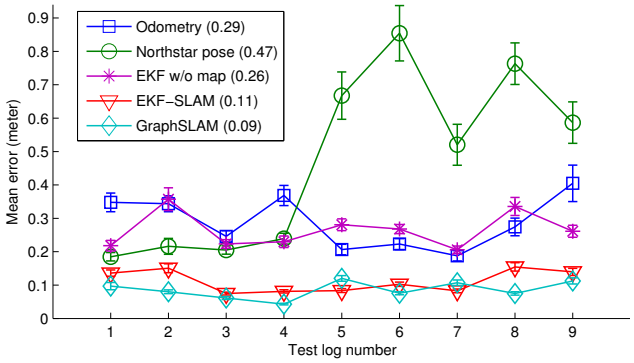


Fig. 13. Mean position errors over all 9 runs with average in parentheses.

VI. Conclusion

We have presented a new approach to localization that learns a vector field of time-stationary signals in an environment. Our method applies as long as the signals change continuously over space, and the vector of signals provides enough information to estimate pose.

The grid resolution of the vector field defines both the approximation error to the true signal curves and the ability of the system to close loops. Since the selection of nodes in the observation model depends on the current pose estimate, a larger cell size increases the chances of choosing the right nodes, i.e. successfully close a loop. A more general model of (23) defined over all nodes, e.g. weighting cells by the mass of pose probability falling into them, might put less emphasis on correct node selection.

Our future work addresses online SLAM methods that are more efficient than the quadratic time complexity of EKF-SLAM and also have smaller storage requirements. These methods allow Vector Field SLAM to run on embedded systems such as those used in robotic floor cleaners.

References

[1] J. Borenstein, J.B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A.K. Peters, Ltd., Wellesley, MA, 1996.

[2] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6), 2007.

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

[4] 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*, 24:381, 1981.

[5] 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.

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

[8] 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.

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

[10] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs *et al.*, editor, *Vision Algorithms: Theory & Practice*. Springer-Verlag, 2000.

[11] 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.