A Marginalized Particle Filtering Framework for Simultaneous Localization and Mapping

Thomas B. Schön, Rickard Karlsson, David Törnqvist, and Fredrik Gustafsson
Division of Automatic Control
Department of Electrical Engineering
Linköpings universitet
SE–581 83 Linköping, Sweden
Email: {schon, rickard, tornqvist, fredrik}@isy.liu.se

Abstract— **This contribution aims at unifying two recent trends in applied particle filtering (PF). The first trend is the major impact in simultaneous localization and mapping (SLAM) applications, utilizing the FastSLAM algorithm. The second one is the implications of the marginalized particle filter (MPF) or the Rao-Blackwellized particle filter in positioning and tracking applications. An algorithm is introduced, which merges FastSLAM and MPF, and the result is an MPF algorithm for SLAM applications, where state vectors of higher dimensions can be used. Results using experimental data from a 3D SLAM development environment, fusing measurements from inertial sensors (accelerometer and gyroscopes) and vision are presented.**

Keywords: Marginalized Particle Filter, Kalman filter, sensor fusion, nonlinear estimation, simultaneous localization and mapping, inertial sensors, vision.

I. Introduction

The main task in localization/positioning and tracking is to estimate for instance the position and orientation of the object under consideration. The particle filter, [1], [2], has proven to be an enabling technology for many applications of this kind, in particular when the observations are complicated nonlinear functions of the position and heading [3]. Furthermore, the *marginalized particle filter* (also known as the Rao-Blackwellized particle filter) [4]–[9] enables estimation of velocity, acceleration, and sensor error models by utilizing any linear Gaussian sub-structure in the model, which is fundamental for performance in applications as surveyed in [10].

The MPF as described in [9] splits the state vector x into two parts, one part x_t^p which is estimated using the particle filter and another part x_t^k where Kalman filters are applied. Basically, the MPF makes use of the following factorization of the posterior distribution of the state vector, which follows from Bayes' rule,

$$p(x_{1:t}^p, x_t^k|y_{1:t}) = p(x_t^k|x_{1:t}^p, y_{1:t})p(x_{1:t}^p|y_{1:t}), \qquad (1)$$

where $y_{1:t} \triangleq \{y_1, \ldots, y_t\}$ denotes the measurements up to time t. If the model is conditionally linear Gaussian, i.e., if the term $p(x_t^k|x_{1:t}^p, y_{1:t})$ is linear Gaussian, it can be optimally estimated using the Kalman filter, whereas for the second factor we have to resort to the PF.

Simultaneous localization and mapping (SLAM) is an extension of the localization or positioning problem to the case where the environment is un-modeled and has to be mapped on-line. An introduction to the SLAM problem is given in the survey papers [11], [12] and the recent book [13]. Many researchers, including us, make use of several sensors in order to tackle the SLAM problem. However, there is also interesting work by e.g., [14]–[17] using only vision.

The FastSLAM algorithm introduced in [18] has proven to be an enabling technology for such applications. FastSLAM can be seen as a special case of MPF, where the map state m_t containing the positions for all landmarks used in the mapping can be interpreted as the linear Gaussian state above. The main difference is that the map vector is a constant parameter with a dimension increasing over time, rather than a time-varying state with a dynamic evolution over time. The derivation is completely analogous to (1), and makes use of the following factorization

$$p(x_{1:t}, m_t|y_{1:t}) = p(m_t|x_{1:t}, y_{1:t})p(x_{1:t}|y_{1:t}). \qquad (2)$$

The FastSLAM algorithm was originally devised to solve the SLAM problem for mobile robots, where the dimension of the state vector is small, typically consisting of three states (2D position and a heading angle) [13]. This implies that all platform states can be estimated by the PF.

Parallelling the evolution of PF applications to high dimensional state vectors, the aim of this contribution is to unify the ideas presented in [9], [19] in order to extend the FastSLAM [18] algorithm to be able to cope with high dimensional state vectors as well. Basically, the main result follows from

$$\begin{aligned} &p(x_{1:t}^p, x_t^k, m_t|y_{1:t}) \\ &\quad = p(m_t|x_t^k, x_{1:t}^p, y_{1:t})p(x_t^k|x_{1:t}^p, y_{1:t})p(x_{1:t}^p|y_{1:t}). \end{aligned} \qquad (3)$$

The derived algorithm is applied to experimental data from a development environment tailored to provide accurate values of ground truth. Here, a high precision industrial robot is

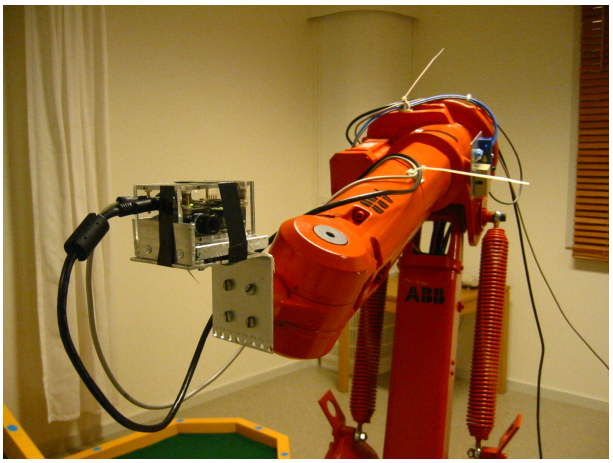

Fig. 1. The 6 DoF ABB IRB1400 industrial robot equipped with the custom made sensor.

programmed to move, possibly using its 6 degree-of-freedom, while the sensor unit, consisting of 3D accelerometers, 3D gyroscopes, and a camera is attached to it. See Fig. 1 for the experimental setup. This allows us to perform repeatable experiments with access to the ground truth (from the industrial robot), so the performance of the algorithm can be accurately assessed.

In Section II the problem under consideration is formulated in more detail. The proposed algorithm is given and explained in Section III. This algorithm is then applied to an application example in Section IV. Finally, the conclusions are given in Section V.

II. Problem Formulation

The aim of this work is, as was explained in the introduction, to solve the SLAM problem when the state dimension of the platform is to large to be estimated by the PF. This section provides a more precise problem formulation and introduces the necessary notation.

The total state vector to be estimated at time t is

$$x_t = \begin{pmatrix} (x_t^p)^T & (x_t^k)^T & m_t^T \end{pmatrix}^T, \tag{4}$$

where x_t^p denotes the states of the platform that are estimated by the particle filter and x_t^k denotes the states of the platform that are marginalized together with the map states m_t. These marginalized states are estimated using (extended) Kalman filters. The map states m_t consists of the entire map at time t, i.e.,

$$m_t = \begin{pmatrix} m_{1,t}^T & \dots & m_{M_t,t}^T \end{pmatrix}^T, \tag{5}$$

where $m_{j,t}$ denotes the position of the j^{th} map entry and M_t denotes the number of entries in the map at time t.

The aim of this work can be formalized as trying to estimate the following filtering PDF,

$$p(x_{1:t}^p, x_t^k, m_t | y_{1:t}). \tag{6}$$

In other words, we are trying to solve the nonlinear filtering problem, providing an estimate of (6).

The *key* factorization, which allows us to solve this problem successfully is

$$\begin{aligned} &p(x_{1:t}^p, x_t^k, m_t | y_{1:t}) \\ &= \underbrace{\prod_{j=1}^{M_t} p(m_{j,t} | x_{1:t}^p, x_t^k, y_{1:t}) p(x_t^k | x_{1:t}^p, y_{1:t})}_{\text{(extended) Kalman filter}} \underbrace{p(x_{1:t}^p | y_{1:t})}_{\text{particle filter}} \end{aligned} \tag{7}$$

In order to devise an estimator for (6) a model describing both the dynamics of the platform and how the measurements obtained from the senors are related to the states are needed. We want a general algorithm, which is applicable to many different platforms (aircraft, helicopters, cars, etc.). Hence, a general model and a rather general model structure is needed, where the algorithm is applicable is given by,

$$x_{t+1}^p = f_t^p(x_t^p) + A_t^p(x_t^p) x_t^k + G_t^p(x_t^p) w_t^p, \tag{8a}$$
$$x_{t+1}^k = f_t^k(x_t^p) + A_t^k(x_t^p) x_t^k + G_t^k(x_t^p) w_t^k, \tag{8b}$$
$$m_{j,t+1} = m_{i,t}, \qquad j = 1, \dots, M_t, \tag{8c}$$
$$y_{1,t} = h_{1,t}(x_t^p) + C_t(x_t^p) x_t^k + e_{1,t}, \tag{8d}$$
$$y_{2,t}^{(j)} = h_{2,t}(x_t^p, m_{j,t}) + e_{2,t}^{(j)}, \tag{8e}$$

where the noise for the platform states is assumed white and Gaussian distributed with

$$w_t = \begin{pmatrix} w_t^p \\ w_t^k \end{pmatrix} \sim \mathcal{N}(0, Q_t), \quad Q_t = \begin{pmatrix} Q_t^p & Q_t^{pk} \\ (Q_t^{pk})^T & Q_t^k \end{pmatrix}. \tag{8f}$$

The measurement noise is assumed white and Gaussian distributed according to

$$e_{1,t} \sim \mathcal{N}(0, R_{1,t}), \tag{8g}$$
$$e_{2,t}^{(j)} \sim \mathcal{N}(0, R_{2,t}), \qquad j = 1, \dots, M_t. \tag{8h}$$

Finally, x_0^k is Gaussian,

$$x_0^k \sim \mathcal{N}(\bar{x}_0, \bar{P}_0), \tag{8i}$$

and the density for x_0^p can be arbitrary, but it is assumed known.

There are two different measurement models, (8d) and (8e), where the former only measures quantities related to the platform, whereas the latter will also involve the map states. Section IV describes a detailed application example using experimental data, where (8d) is used to model inertial sensors and (8e) is used to model a camera.

III. Marginalized Particle Filter SLAM

In Section III-A the MPF SLAM algorithm is given and in the subsequent sections the details of the algorithm are discussed in more detail.

A. Algorithm

The algorithm presented in this paper draws on several rather well known algorithms and it is based on the marginalized particle filter, [4]–[9]. The FastSLAM algorithm, [18], is extended by not only marginalizing the map states, but also the states corresponding to a linear Gaussian sub-structure present in the model for the platform, as was explained in Section I and Section II. Assuming that the platform is modeled in the form given in (8), the MPF-SLAM method is given in Algorithm 1.

Algorithm 1: **MPF–SLAM**

1) Initialize the particles
$$x_{1|0}^{p,(i)} \sim p(x_{1|0}^{p}),$$
$$x_{1|0}^{k,(i)} = \bar{x}_{1|0}^{k},$$
$$P_{1|0}^{k,(i)} = \bar{P}_{1|0}, \qquad i = 1, \ldots, N,$$
where N denotes the number of particles.
2) If there is a new map related measurement available perform data association for each particle, otherwise proceed to step 3.
3) Compute the importance weights according to
$$\gamma_t^{(i)} = p(y_t | x_{1:t}^{p,(i)}, y_{1:t-1}), \qquad i = 1, \ldots, N,$$
and normalize $\tilde{\gamma}_t^{(i)} = \gamma_t^{(i)} / \sum_{j=1}^{N} \gamma_t^{(j)}$.
4) Draw N new particles with replacement (resampling) according to
$$Pr(x_{t|t}^{(i)} = x_{t|t}^{(j)}) = \tilde{\gamma}_t^{(j)}, i = 1, \ldots, N.$$
5) If there is a new map related measurement perform map estimation and management (detailed below), otherwise proceed to step 6.
6) Particle filter prediction and Kalman filter (for each particle $i = 1, \ldots, N$)
   a) Kalman filter measurement update,
   $$p(x_t^k | x_{1:t}^p, y_{1:t}) = \mathcal{N}(x_t^k | \hat{x}_{t|t}^{k,(i)}, P_{t|t}^{(i)}),$$
   where $\hat{x}_{t|t}^{k,(i)}$ and $P_{t|t}^{(i)}$ are given in (11).
   b) Time update for the nonlinear particles,
   $$x_{t+1|t}^{p,(i)} \sim p(x_{t+1|t} | x_{1:t}^{p,(i)}, y_{1:t}).$$
   c) Kalman filter time update,
   $$p(x_{t+1}^k | x_{1:t+1}^p, y_{1:t}) = \mathcal{N}(\hat{x}_{t+1|t}^k | x_{1:t}^{p,(i)}, P_{t+1|t}^{(i)}),$$
   where $\hat{x}_{t+1|t}^{k,(i)}$ and $P_{t+1|t}^{(i)}$ are given by (12).
7) Set $t := t + 1$ and iterate from step 2.

Note that y_t denotes the measurements present at time t. The following theorem will give all the details for how to compute the Kalman filtering quantities. It is important to stress that all available embellishments available for that particle filter can be used together with Algorithm 1. To give one example, the so called FastSLAM 2.0 makes use of an improved importance function in step 6b [20].

Theorem 1: Using the model given by (8), the conditional probability density functions for x_t^k and x_{t+1}^k are given by
$$p(x_t^k | x_{1:t}^p, y_{1:t}) = \mathcal{N}(\hat{x}_{t|t}^k, P_{t|t}), \tag{10a}$$
$$p(x_{t+1}^k | x_{1:t+1}^p, y_{1:t}) = \mathcal{N}(\hat{x}_{t+1|t}^k, P_{t+1|t}), \tag{10b}$$
where
$$\hat{x}_{t|t}^k = \hat{x}_{t|t-1}^k + K_t(y_{1,t} - h_{1,t} - C_t \hat{x}_{t|t-1}^k), \tag{11a}$$
$$P_{t|t} = P_{t|t-1} - K_t S_{1,t} K_t^T, \tag{11b}$$
$$S_{1,t} = C_t P_{t|t-1} C_t^T + R_{1,t}, \tag{11c}$$
$$K_t = P_{t|t-1} C_t^T S_{1,t}^{-1}, \tag{11d}$$
and
$$\hat{x}_{t+1|t}^k = \bar{A}_t^k \hat{x}_{t|t}^k + G_t^k (Q_t^{kp})^T (G_t^p Q_t^p)^{-1} z_t + f_t^k + L_t(z_t - A_t^p \hat{x}_{t|t}^k), \tag{12a}$$
$$P_{t+1|t} = \bar{A}_t^k P_{t|t} (\bar{A}_t^k)^T + G_t^k \bar{Q}_t^k (G_t^k)^T - L_t N_t L_t^T, \tag{12b}$$
$$S_{2,t} = A_t^p P_{t|t} (A_t^p)^T + G_t^p Q_t^p (G_t^p)^T, \tag{12c}$$
$$L_t = \bar{A}_t^k P_{t|t} (A_t^p)^T S_{2,t}^{-1}, \tag{12d}$$
where
$$z_t = x_{t+1}^p - f_t^p, \tag{13a}$$
$$\bar{A}_t^k = A_t^k - G_t^k (Q_t^{kp})^T (G_t^p Q_t^p)^{-1} A_t^p, \tag{13b}$$
$$\bar{Q}_t^k = Q_t^k - (Q_t^{kp})^T (Q_t^p)^{-1} Q_t^{kp}. \tag{13c}$$
Proof: See [9]. ∎

B. Data Association

Data association is a complicated problem, but it has been studied extensively in the literature for many tracking applications, [21]–[23]. Classical methods such as the *nearest neighbor* (NN), *probabilistic data association* (PDA), *joint probabilistic data association* (JPDA) or *multi-hypothesis tracking* (MHT) exist for single and multiple targets. Depending on the number of targets, the false alarm rate and the probability of detection these methods varies in performance and ability to express these phenomena. The methods mentioned above were originally developed to use together with estimators based on Kalman filters. The use of particle filters opens up for other data association methods, typically more integrated with the filter.

The MPF implementation of the SLAM problem will lead to an increased complexity for the data association. This is because for each particle in the MPF there exist several map entries. Hence, many classical association methods will be too computationally intensive for a direct implementation.

C. Likelihood Computation

In order to compute the importance weights $\{\gamma_t^{(i)}\}_{i=1}^N$ in Algorithm 1, the following likelihoods have to be evaluated
$$\gamma_t^{(i)} = p(y_t | x_{1:t}^{p,(i)}, y_{1:t-1}), \qquad i = 1, \ldots, N. \tag{14}$$

The standard way of performing this type of computation is simply to marginalize the Kalman filter variables x_t^k and $\{m_{j,t}\}_{j=1}^{M_t}$,

$$p(y_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = \int p(y_t, x_t^k, m_t|x_{1:t}^{p,(i)}, y_{1:t-1}) dx_t^k dm_t, \tag{15}$$

where

$$p(y_t, x_t^k, m_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = p(y_t|x_t^k, m_t, x_t^p) \times p(x_t^k|x_{1:t}^p, y_{1:t-1}) \prod_{j=1}^{M_t} p(m_{j,t}|x_{1:t}^p, y_{1:t-1}). \tag{16}$$

Let us consider the case where both $y_{1,t}$ and $y_{2,t}$ are present. Note that the cases where either $y_{1,t}$ or $y_{2,t}$ are present are just special cases of this. First of all, the measurements are conditionally independent given the state, implying that

$$p(y_t|x_t^k, m_t, x_t^p) = p(y_{1,t}|x_t^k, x_t^p) \prod_{j=1}^{M_t} p(y_{2,t}^{(j)}|x_t^p). \tag{17}$$

Now, inserting (17) into (16) gives

$$p(y_t, x_t^k, m_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = p(y_{1,t}|x_t^k, x_t^p) \times p(x_t^k|x_{1:t}^p, y_{1:t-1}) \prod_{i=1}^{M_t} p(m_{i,t}|x_{1:t}^p, y_{1:t-1}) p(y_{2,t}^{(i)}|x_t^p), \tag{18}$$

which inserted in (15) finally results in

$$p(y_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = \int p(y_{1,t}|x_t^k, x_t^p) p(x_t^k|x_{1:t}^p, y_{1:t-1}) dx_t^k \times \prod_{j=1}^{M_t} \int p(y_{2,t}^{(j)}|x_t^p) p(m_{j,t}|x_{1:t}^p, y_{1:t-1}) dm_{1,t} \cdots dm_{M_t,t}. \tag{19}$$

All the densities present in (19) are known according to

$$p(x_t^k|x_{1:t}^p, y_{1:t-1}) = \mathcal{N}(x_t^k|\hat{x}_{t|t-1}^k, P_{t|t-1}), \tag{20a}$$
$$p(m_{j,t}|x_{1:t}^p, y_{1:t-1}) = \mathcal{N}(m_t|\hat{m}_{j,t-1}, \Sigma_{t-1}), \tag{20b}$$
$$p(y_{1,t}|x_t^k, x_t^p) = \mathcal{N}(y_{1,t}|h_{1,t}(x_t^p) + C_t(x_t^p)x_t^k, R_1), \tag{20c}$$
$$p(y_{2,t}^{(j)}|x_t^p) = \mathcal{N}(y_{2,t}^{(j)}|h_{2,t}(x_t^p, \hat{m}_{j,t}) + H_{j,t}m_{j,t}, R_2). \tag{20d}$$

Here it is important to note that the standard FastSLAM approximation has been invoked in order to obtain (20d). That is, the measurement equation (8e) is linearized with respect to the map states $m_{j,t}$. The reason for this approximation is that we have to marginalize all the map states, otherwise the dimension will be much too large for the particle filter to handle. Using (20), the integrals in (19) can now be solved, resulting in

$$p(y_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = \mathcal{N}(y_{1,t} - h_{1,t} - C_t\hat{x}_{t|t-1}^k, C_t P_{t|t-1} C_t^T) \times \prod_{j=1}^{M_t} \mathcal{N}(y_{2,t}^{(j)} - h_{2,t}^{(j)} - H_{j,t}\hat{m}_{j,t-1}, H_{j,t}\Sigma_{t-1}(H_{j,t})^T + R_2) \tag{21}$$

D. Map Estimation and Map Management

A simple map consists of a collection of map entries $\{m_{j,t}\}_{j=1}^{M_t}$, each consisting of

- $\hat{m}_{j,t}$ – estimate of the position (three dimensions).
- $\Sigma_{j,t}$ – covariance for the position estimate.

Note that this is probably the most simple map possible. Each particle has an entire map estimate associated to it. Step 5 of Algorithm 1 consists of updating these map estimates in accordance with the new map related measurements that are available. First of all, if a measurement has been successfully associated to a certain map entry, it is updated using the standard Kalman filter measurement update according to

$$m_{j,t} = m_{j,t-1} + K_{j,t}\left(y_{2,t}^{(j)} - h_t^{(j)}\right), \tag{22a}$$
$$\Sigma_{j,t} = \left(I - K_{j,t}H_{i,t}^T\right)\Sigma_{j,t-1}, \tag{22b}$$
$$K_{j,t} = \Sigma_{j,t-1}H_{j,t}\left(H_{j,t}\Sigma_{j,t-1}H_{j,t}^T + R_2\right)^{-1}. \tag{22c}$$

If an existing map entry is not observed the corresponding map estimate is simply propagated according to its dynamics, i.e., it is unchanged

$$m_{j,t} = m_{j,t-1}, \tag{23a}$$
$$\Sigma_{j,t} = \Sigma_{j,t-1}. \tag{23b}$$

It has to be investigated if any of the entries should be removed from the map.

Finally, initialization of new map entries have to be handled. If $h_{2,t}(x_t^p, m_{j,t})$ is invertible this can be used to directly initialize the position from the measurement $y_{2,t}$. However, $h_{2,t}(x_t^p, m_{j,t})$ is typically not invertible, implying that there is a need for more than one measurement in order to be able to initialize a new map entry, utilizing for example triangulation.

E. Approximate Algorithm

The computational complexity of Algorithm 1 will inevitably be high, implying that it is interesting to consider any ideas on how to reduce it. In the present context multi-rate sensors (sensors providing measurements with different sampling times) are typically used. This can be exploited, similarly to [19]. The idea is very simple, rather than running Algorithm 1 at the same frequency as the fast sensor, it is executed at the same frequency as the slow sensor, with a nested filter handling the fast sensor measurements. Obviously, this will involve approximations.

In this way both the linear Gaussian sub-structure in the platform model and the multi-rate properties of the sensors are exploited in order to obtain an efficient algorithm [19].

IV. Application Example

In this section we provide a rather detailed treatment of a SLAM application, where Algorithm 1 is used to fuse measurements from a camera, three accelerometers and three gyroscopes. The sensor has been attached to a high precision

6 DoF ABB IRB1440 industrial robot. The reason for this is that the robot will allow us to make repeatable 6 DoF motions and provide the true position and orientation with very high accuracy, enabling systematic evaluation of algorithms. The sensor and its mounting to the industrial robot is illustrated in Fig. 1. A detailed picture of the sensor is provided on page 9 in [24].

The main objective is to find the position and orientation of the sensor from sensor data only, despite problems such as biases in the measurements. In the surrounding neighbourhood features are placed in such a way that the vision system can easily extract features. These are used in SLAM to reduce the problem caused by inertial drift and bias in the IMU sensor.

A. Model

The kinematic part of the state vector consists of position $p_t \in \mathbb{R}^3$, velocity $v_t \in \mathbb{R}^3$, and acceleration $a_t \in \mathbb{R}^3$, all described in an earth-fixed reference frame. Furthermore, the state vector is extended with bias states for acceleration $b_{a,t} \in \mathbb{R}^3$, and angular velocity $b_{\omega,t} \in \mathbb{R}^3$ in order to account for sensor imperfections. The state vector also contains the angular velocity ω_t and a unit quaternion q_t, which is used to parameterize the rotation group $SO(3)$.

In order to put the model in the MPF framework, the state vector is split into two parts, one estimated using Kalman filters x_t^k and one estimated using the particle filter x_t^p. Hence, define

$$x_t^k = \begin{pmatrix} v_t^T & a_t^T & (b_{\omega,t})^T & (b_{a,t})^T & \omega_t^T \end{pmatrix}^T \tag{24a}$$

$$x_t^p = \begin{pmatrix} p_t^T & q_t^T \end{pmatrix}^T, \tag{24b}$$

which means $x_t^k \in \mathbb{R}^{15}$ and $x_t^p \in \mathbb{R}^7$. In inertial estimation it is essential to clearly state which coordinate system any entity is expressed in. Here the notation is simplified by suppressing the coordinate system for the earth-fixed states, which means that

$$p_t = p_t^e, \quad v_t = v_t^e, \quad a_t = a_t^e, \tag{25a}$$

$$w_t = w_t^e, \quad b_{\omega,t} = b_{\omega,t}^e, b_{a,t} = b_{a,t}^e. \tag{25b}$$

Likewise, the unit quaternions represent the rotation from the earth-fixed system to the body (IMU) system, since the IMU is rigidly attached to the body.

$$q_t = q_t^{be} = \begin{pmatrix} q_{t,0} & q_{t,1} & q_{t,2} & q_{t,3} \end{pmatrix}^T \tag{26}$$

The quaternion estimates are normalized, to make sure that they still parameterizes a rotation. Further details regarding rotations are given in Appendix I.

1) Dynamic Model: The dynamic model describes how the platform and the map entries evolve over time. These equations are given below, in the form (8a) – (8d), suitable for direct use in Algorithm 1.

$$\underbrace{\begin{pmatrix} v_{t+1} \\ a_{t+1} \\ b_{\omega,t+1} \\ b_{a,t+1} \\ \omega_{t+1} \end{pmatrix}}_{x_{t+1}^k} = \underbrace{\begin{pmatrix} I & TI & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{pmatrix}}_{A_t^k} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x_t^k} + \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}}_{G_t^k} \underbrace{\begin{pmatrix} w_{1,t} \\ w_{2,t} \\ w_{3,t} \\ w_{4,t} \end{pmatrix}}_{w_t^k} \tag{27a}$$

$$\underbrace{\begin{pmatrix} p_{t+1} \\ q_{t+1} \end{pmatrix}}_{x_{t+1}^p} = \underbrace{\begin{pmatrix} p_t \\ q_t \end{pmatrix}}_{f_t^p(x_t^p)} + \underbrace{\begin{pmatrix} TI & \frac{T^2}{2} I & 0_{3\times 9} \\ 0_{4\times 3} & 0_{4\times 9} & -\frac{T}{2}\tilde{S}(q) \end{pmatrix}}_{A_t^p(x_t^p)} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x_t^k}, \tag{27b}$$

$$m_{j,t+1} = m_{j,t}, \qquad j = 1, \ldots, M_t, \tag{27c}$$

where

$$\tilde{S}(q) = \begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{pmatrix}, \tag{28}$$

and where I denotes the 3×3 unit matrix and 0 denotes the 3×3 zero matrix. The process noise w_t^k is assumed to be independent and Gaussian, with covariance $Q_t^k = (Q_a \oplus Q_{b_\omega} \oplus Q_{b_a} \oplus Q_\omega)$, where $\oplus$ denotes that direct sum.

2) Measurement Model – Inertial Sensors: The IMU consists of accelerometers measuring accelerations $y_{a,t}$ in all three dimensions, a gyroscope measuring angular velocities $y_{\omega,t}$ in three dimensions and a magnetometer measuring the direction to the magnetic north pole. Due to the fairly magnetic environment it is just the accelerometers and gyroscopes that are used for positioning. The inertial sensors operate at 100 Hz. For further details on inertial sensors, see for instance [25]–[27]. The inertial measurements relate to the states according to,

$$y_{1,t} = \begin{pmatrix} y_{\omega,t} \\ y_{a,t} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 \\ -R(q_t) g^e \end{pmatrix}}_{h(x_t^p)} + \underbrace{\begin{pmatrix} 0_3 & 0_3 & I & 0_3 & R(q_t) \\ 0_3 & R(q_t) & 0_3 & I & 0_3 \end{pmatrix}}_{C(x_t^p)} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x_t^k} + \underbrace{\begin{pmatrix} e_{1,t} \\ e_{2,t} \end{pmatrix}}_{e_t}, \tag{29}$$

which obviously is in the form required by (8). The measurement noise e_t is assumed Gaussian with covariance $R_t = (R_\omega \oplus R_a)$.

3) Measurement Model – Camera: Before the camera images are used they are adjusted according to the calibration. This allows us to model the camera using the pinhole model with focal length $f = 1$, according to [28], [29],

$$y_{2,t} = y_{m_j,t} = \underbrace{\frac{1}{z_t^c}\begin{pmatrix} x_t^c \\ y_t^c \end{pmatrix}}_{h^c(m_{j,t},p_t,q_t)} + e_{3,t}, \tag{30}$$

where

$$m_{j,t}^c = \begin{pmatrix} x_t^c \\ y_t^c \\ z_t^c \end{pmatrix} = R(q_t^{cb})R(q_t^{be})(m_{j,t} - p_t) + r^c. \tag{31}$$

Here, r^c is a fixed vector representing the translation between the camera and the IMU and q_t^{cb} is the unit quaternion describing the rotation from the IMU to the camera. The covariance for the measurement noise is denoted R_c.

This particular sensor is equipped with an internal camera, which is synchronized in time to the inertial measurements. This provides a good setup for fusing vision information with the kinematic states. Images are available at 12.5 Hz in a resolution of 340×280 pixels. In order to use vision for feature extraction and estimation, we have made use of standard camera calibration techniques, see e.g., [30].

The features are not exactly in the form suitable for marginalization in the particle filter. Hence, we are forced to an approximation in order to obtain a practical algorithm. The standard approximation [13] is in this case simply to linearize the map equations according to,

$$\begin{aligned} y_{m_{j,t}} &= h^c(m_{j,t}, p_t, q_t) + e_{3,t} && (32)\\ &\approx \underbrace{h_j^c(\hat{m}_{j,t|t-1}, p_t, q_t) - H_{j,t}\hat{m}_{j,t|t-1}}_{h(x_t^p)} \\ &\quad + H_{j,t}m_{j,t} + e_{3,t}, \quad j = 1, \ldots, M_t, && (33) \end{aligned}$$

where the Jacobian matrix $H_{i,t}$ is straightforwardly computed using the chain rule, i.e.,

$$H_{i,t} = \frac{\partial h^c}{\partial m_i} = \frac{\partial h^c}{\partial m_i^c}\frac{\partial m_i^c}{\partial m_i}, \tag{34}$$

The two partial derivatives in (34) are given by

$$\frac{\partial h^c}{\partial m_i^c} = \begin{pmatrix} \frac{1}{z^c} & 0 & -\frac{x^c}{(z^c)^2} \\ 0 & \frac{1}{z^c} & -\frac{y^c}{(z^c)^2} \end{pmatrix} \tag{35}$$

$$\frac{\partial m_i^c}{\partial m_i} = R(q_t^{cb})R(q_t^{be}). \tag{36}$$

The camera model delivers point-measurements of features in the field-of-view. This can be done using several different methods with different performance. An often used detector is the Harris detector [31], which basically extracts well-defined corners in an image. It can be extended with a shift invariant version, the so-called *scale invariant feature transform* (SIFT) [32]. It is also possible to refine the feature detection process even further by estimating the slope of an image plane [14]. Many of these have good performance, however at a rather high computational complexity. In the current setup the environment is from a computer vision perspective rather simple, hence faster and simpler corner detectors can be applied.

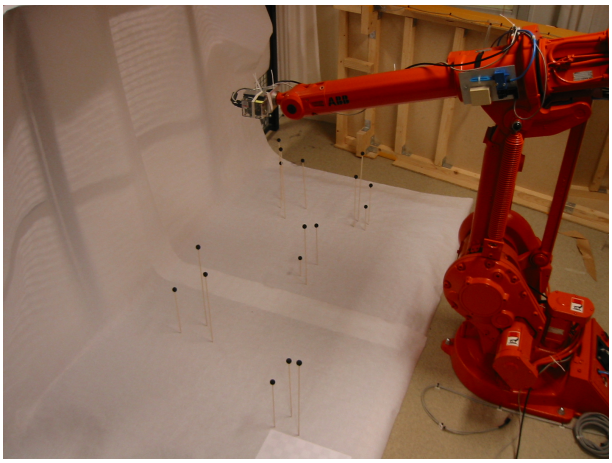

Fig. 2. The SLAM setup in the industrial robot laboratory, with the camera and IMU sensor at the manipulator, and where marks are placed at known locations to emulate features (landmarks).

B. Experiment Setup

The scene surrounding the platform is equipped with easily detected and distinguished features placed in the vicinity as depicted in Fig. 2. In Table I the parameters in the SLAM MPF navigation example are presented.

TABLE I

SYSTEM, FILTER, AND SENSOR PARAMETERS FOR THE EXPERIMENT.

Process Noise	
Cov. Acceleration	$Q_a = diag\begin{pmatrix}0.5^2 & 0.5^2 & 0.5^2\end{pmatrix}$
Cov. Bias angular rate	$Q_{b_\omega} = diag\begin{pmatrix}10^{-10} & 10^{-10} & 10^{-10}\end{pmatrix}$
Cov. Bias acceleration	$Q_{b_a} = diag\begin{pmatrix}0.001^2 & 0.001^2 & 0.001^2\end{pmatrix}$
Cov. Angular rate	$Q_\omega = diag\begin{pmatrix}0.001^2 & 0.001^2 & 0.001^2\end{pmatrix}$
Measurement Noise	
Cov. Camera	$R_c = diag\begin{pmatrix}0.01^2 & 0.01^2\end{pmatrix}$
Cov. Accelerometer	$R_a = diag\begin{pmatrix}0.02^2 & 0.02^2 & 0.03^2\end{pmatrix}$
Cov. Gyroscope	$R_w = diag\begin{pmatrix}0.02^2 & 0.03^2 & 0.03^2\end{pmatrix}$
System	
Sample freq. (IMU)	$1/T = 100$ Hz
Sample freq. camera	$1/T_c = 12.5$ Hz
Camera resolution	340×280 pixels
Number of particles	$N = 100$

In the example presented here, the noise level in the measurements is such that a simplified version of the NN method was used. Hence, the idea is to associate only one measurement to a track (map entry) at any given time. Instead of optimizing globally (*global nearest neighbor* (GNN)), this is done sub-optimally, assigning the closest one first. This seems to work perfectly well for the present application.

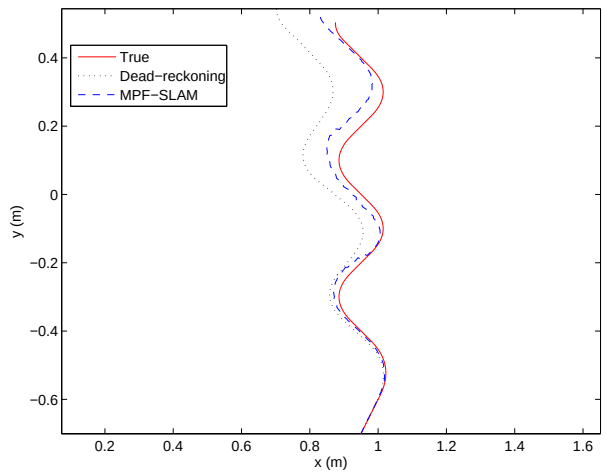

Fig. 3. The ground truth position from the industrial robot, dead-reckoned position from inertial sensor data (5 seconds in 100 Hz), the estimate from the SLAM IMU algorithm with vision measurements.

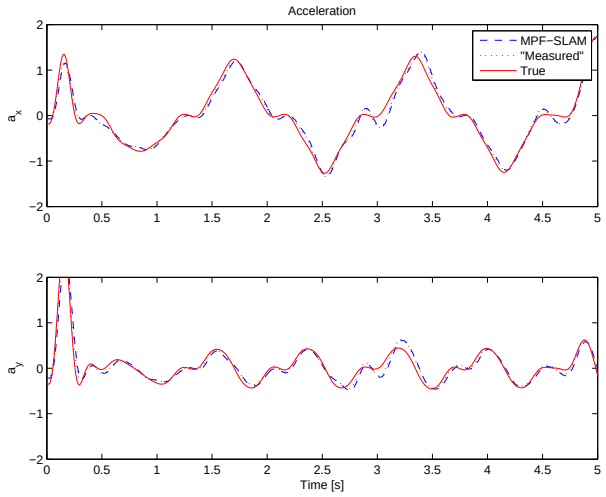

Fig. 4. The true acceleration (from the industrial robot), the measured acceleration (from the IMU sensor) and the result from Algorithm 1.

The experiment begins with zero velocity at a given point. Before starting the experiment the IMU sensor was calibrated, in order to correct for initial bias contribution. In Fig. 3 the true trajectory is depicted. Note here that the motion is only in the horizontal plane, i.e., with a fixed height, and with constant orientation.

C. Experimental Results

In this section results from the experiment are presented. In Fig. 3 the result from dead-reckoning, i.e., double integration of the IMU acceleration data directly (after appropriate rotation from sensor to the earth-fixed system) is depicted together with the estimated trajectory using the MPF SLAM method. The ground truth is evaluated from accurate measurements in the robot and using its known geometry in kinematic calculations. Since the motion was rather smooth, high frequency dynamics in the robot arm was not excited, hence yielding a very accurate ground truth value. As can be seen the performance in position is drastically increased when the MPF SLAM method is used, compared to only dead-reckoning. The reason is the filter, where the orientation and acceleration coupling together with the map features improve the kinematic states. In the current experiment the bias term was not used.

In Fig. 4 the measurement from the accelerometers, the MPF, and the ground truth from the industrial robot are depicted for the experiment.

The vision system is used to build up the map (feature tracks of landmarks), which is depicted in Fig. 5 for a specific time. Note that the uncertainty is visible in the spread of the particle cloud.

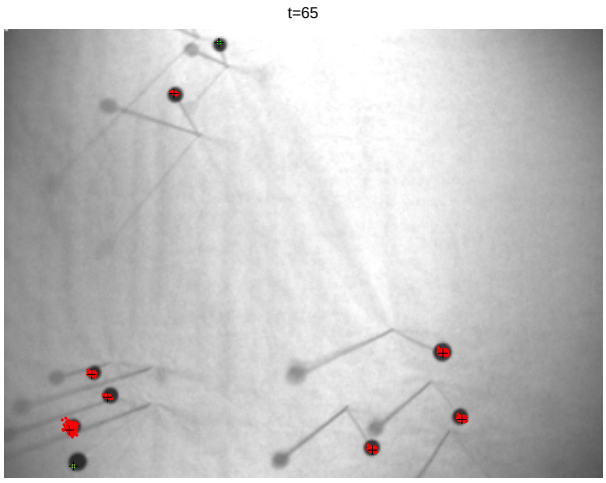

Fig. 5. Map features from the feature extraction (cross) and with the uncertainty represented by the particle filter cloud (red dots). Please note that this figure has to be viewed in color.

V. Conclusion

This contribution has introduced a marginalized particle filtering framework for solving the SLAM problem, enabling more high dimensional state vectors for the platform. The idea draws on the FastSLAM algorithm, but rather than marginalizing only the map, we also marginalize all the states corresponding to a linear Gaussian sub-structure in the model. The approach was validated using a simple application example, where inertial measurements where fused with vision measurements. The correctness of the result was assessed using the ground truth.

Acknowledgment

The authors would like to thank the MOVIII (Modeling, Visualization and Information Integration), the Swedish research council (VR) and the MATRIS project (EU research project, contract number IST-002013) for funding the work.

APPENDIX I
COORDINATE SYSTEMS

The following convention is used to rotate a vector from a coordinate system A to a coordinate system B,

$$x_{\text{B}} = R_{\text{AB}} x_{\text{A}}.$$

where R_{AB} is used to denote the rotation matrix describing the rotation from B to A. Hence, we can get from system A to C, via B according to

$$R_{\text{CA}} = R_{\text{CB}} R_{\text{BA}}.$$

This can also be expressed using unit quaternions q_{A},

$$u_{\text{B}} = (q_{\text{A}})^{\star} \circledast u_{\text{A}} \circledast q_{\text{A}},$$

where u_{A} is the quaternion extension of the vector x_A, i.e., $u_{\text{A}} = \begin{pmatrix} 0 & x_{\text{A}}^T \end{pmatrix}^T$ and $\circledast$ represents quaternion multiplication. Furthermore, $u^{\star}$ denotes the quaternion conjugate. See e.g., [33], [34] for an introduction to unit quaternions and related rotation parameterizations.

It is straightforward to convert a given quaternion into the corresponding rotation matrix,

$$R(q) = \begin{pmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{pmatrix}.$$

The following coordinate systems are used in this paper. An earth-fixed (denoted with e), body or inertial sensor system (denoted with b), and camera system (denoted c).

REFERENCES

[1] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings on Radar and Signal Processing*, vol. 140, 1993, pp. 107–113.

[2] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. New York, USA: Springer Verlag, 2001.

[3] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, Feb. 2002.

[4] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[5] G. Casella and C. P. Robert, "Rao-Blackwellisation of sampling schemes," *Biometrika*, vol. 83, no. 1, pp. 81–94, 1996.

[6] A. Doucet, N. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump Markov linear systems," *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 613–624, 2001.

[7] R. Chen and J. S. Liu, "Mixture Kalman filters," *Journal of the Royal Statistical Society*, vol. 62, no. 3, pp. 493–508, 2000.

[8] C. Andrieu and A. Doucet, "Particle filtering for partially observed Gaussian state space models," *Journal of the Royal Statistical Society*, vol. 64, no. 4, pp. 827–836, 2002.

[9] T. Schön, F. Gustafsson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2279–2289, July 2005.

[10] T. B. Schön, R. Karlsson, and F. Gustafsson, "The marginalized particle filter in practice," in *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2006.

[11] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, Sept. 2006.

[12] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, June 2006.

[13] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents. Cambridge, MA, USA: The MIT Press, 2005.

[14] A. J. Davison, I. Reid, N. Molton, and O. Strasse, "MonoSLAM: Real-time single camera SLAM," *Accepted for publication in IEEE Transactions on Patterns Analysis and Machine Intelligence*, 2007.

[15] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings Ninth IEEE International Conference on Computer Vision*, vol. 2, Nice, France, Oct. 2003, pp. 1403–1410.

[16] A. J. Davison, Y. G. Cid, and N. Kita, "Real-time 3D SLAM with wide-angle vision," in *Proceedings of the 5th IFAC/EUCON Symposium on Intelligent Autonomus Vehicles*, Lisaboa, Portugal, July 2004.

[17] E. Eade and T. Drummond, "Scalable monocular SLAM," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, NY, USA, June 2006, pp. 469–476.

[18] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM a factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Comference on Artificial Intelligence*, Edmonton, Canada, 2002.

[19] T. B. Schön, D. Törnqvist, and F. Gustafsson, "Fast particle filters for multi-rate sensors," in *submitted to 15th European Signal Processing Conference (EUSIPCO)*, Poznan', Poland, Sept. 2007.

[20] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[21] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, ser. Mathematics in science and engineering. Orlando, FL, USA: Academic Press, 1988.

[22] Y. Bar-Shalom and X.-R. Li, *Estimation and Tracking: Principles, Techniques, and Software*. Norwood, MA, USA: Artech House, 1993.

[23] S. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA, USA: Artech House, Inc., 1999.

[24] T. B. Schön, "Estimation of nonlinear dynamic systems – theory and applications," PhD thesis No 998, Linköping Studies in Science and Technology, Department of Electrical Engineering, Linköping University, Sweden, Feb. 2006.

[25] D. H. Titterton and J. L. Weston, *Strapdown inertial navigation technology*, ser. IEE radar, sonar, navigation and avionics series. Stevenage, UK: Peter Peregrinus Ltd., 1997.

[26] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global Positioning Systems, Inertial Navigation, and Integration*. New York, USA: John Wiley & Sons, 2001.

[27] A. Chatfield, *Fundamentals of High Accuracy Inertial Navigation*, 3rd ed. USA: American Institute of Aeronautics and Astronautics, 1997, vol. 174.

[28] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-D vision – from images to geometric models*, ser. Interdisciplinary Applied Mathematics. Springer, 2006.

[29] R. Hartley and A. Zisserman, *Multiple View Geometry in computer vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2003.

[30] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[31] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, Manchester, UK, 1988, pp. 147–151.

[32] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[33] M. D. Shuster, "A survey of attitude representations," *The Journal of the Austronautical Sciences*, vol. 41, no. 4, pp. 439–517, Oct. 1993.

[34] J. B. Kuipers, *Quaternions and Rotation Sequences*. Princeton University Press, 1999.