A Framework for Simultaneous Localization and Mapping Utilizing Model Structure

Thomas B. Schön, Rickard Karlsson, David Törnqvist, and Fredrik Gustafsson
Division of Automatic Control
Department of Electrical Engineering
Linköping University
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 (RBPF) 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 gyro) and vision are presented.**

Keywords: **Rao-Blackwellized/marginalized particle filter, sensor fusion, 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* (PF), [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 *Rao-Blackwellized particle filter* (RBPF) also denoted the *marginalized particle filter* (MPF), [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]. As described in [9], the MPF splits the state vector x_t 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, it uses 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]. From a sensor point of view, there are two ways of tackling this problem. The first way is to use only one sensor, such as vision, see e.g., [14]–[17] and the second way is to fuse measurements from several sensors. This work considers the latter.

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 RBPF/MPF, where the map state m_t, containing the positions for all landmarks used in the mapping, can be interpreted as a linear Gaussian state. 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

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

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 programmed to move, possibly using its 6 *degrees-of-freedom*
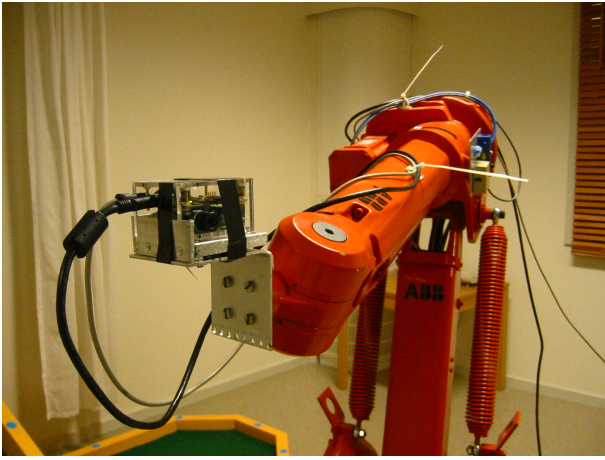
Figure 1. The 6 DoF ABB IRB1400 industrial robot equipped with the custom made sensor.

(DoF), while the sensor unit, consisting of three accelerometers, three gyroscopes, and a camera is attached to it. See Figure. 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 to solve the SLAM problem when the state dimension of the platform is too 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

$$x_t = \left((x_t^p)^T \quad (x_t^k)^T \quad m_t^T \right)^T, \tag{4}$$

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 linear-Gaussian given information about x_t^p. These states together with the map m_t are estimated using Kalman filters. The map states m_t consists of the entire map at time t, i.e.,

$$m_t = \left(m_{1,t}^T \quad \dots \quad m_{M_t,t}^T \right)^T, \tag{5}$$

where $m_{j,t}$ denotes the position of the jth 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 *probability density function* (PDF),

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

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

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

In order to devise an estimator for (6) a *system model* and a *measurement model* are needed. The former describes the dynamic behaviour of the platform, that is how the state x_t evolves over time. The measurement model describes the sensors, i.e., it consists of equations relating the measurements y_t to the state x_t. We want a general algorithm, which is applicable to many different platforms (aircraft, helicopters, cars, etc.). Hence, the model structure should be as general as possible, leading us to,

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

where $j = 1, \dots, M_t$ and the noise for the platform states is assumed white and Gaussian distributed with

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

To simplify the notation in the rest of the paper, denote $f_t^p(x_t^p)$ with f_t^p, $A_t^p(x_t^p)$ with A_t^p and so on. The measurement noise is assumed white and Gaussian distributed according to

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

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. PARTICLE FILTER FOR SLAM UTILIZING STRUCTURE

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

A. Algorithm

The algorithm presented in this paper draws on several rather well known algorithms. It is based on the RBPF/MPF method, [4]–[9]. The FastSLAM algorithm [18] is extended by not only including the map states, but also the states corresponding to a linear Gaussian sub-structure present in the model for the platform. Assuming that the platform is modeled in the form given in (8), the SLAM-method utilizing structure is given in Algorithm 1.

Algorithm 1: *Particle filter for* SLAM *utilizing structure*

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, for each $i = 1, \ldots, N$

$$\Pr(x_{t|t}^{(i)} = x_{t|t}^{(j)}) = \tilde{\gamma}_t^{(j)}, j = 1, \ldots, N.$$

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,

$$p(x_{t+1}^k | x_{1:t+1}^p, y_{1:t})$$
$$= \mathcal{N}(x_{t+1|t}^k | \hat{x}_{t+1|t}^{k,(i)}, P_{t+1|t}^{(i)}),$$

      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 embellishments available for the particle filter can be used together with Algorithm 1. To give one example, the so-called FastSLAM 2.0 makes use of an improved proposal distribution 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

$$\hat{x}_{t+1|t}^k = \bar{A}_t^k \hat{x}_{t|t}^k + G_t^k(Q_t^{kp})^T(G_t^p Q_t^p)^{-1} z_t$$
$$+ f_t^k + L_t(z_t - A_t^p \hat{x}_{t|t}^k), \tag{12a}$$
$$P_{t+1|t} = \bar{A}_t^k P_{t|t}(\bar{A}_t^k)^T + G_t^k \bar{Q}_t^k(G_t^k)^T - L_t S_{2,t} L_t^T, \tag{12b}$$
$$S_{2,t} = A_t^p P_{t|t}(A_t^p)^T + G_t^p Q_t^p(G_t^p)^T, \tag{12c}$$
$$L_t = \bar{A}_t^k P_{t|t}(A_t^p)^T S_{2,t}^{-1}, \tag{12d}$$

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, see e.g., [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 be used 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 particle implementation of the SLAM problem will lead to an increased complexity for the data association. This is because for each particle in the filter 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

$$\gamma_t^{(i)} = p(y_t|x_{1:t}^{p,(i)}, y_{1:t-1}), \qquad i = 1, \ldots, N. \qquad (14)$$

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

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

where

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

Let us consider the case where both $y_{1,t}$ and $y_{2,t}$ are present, i.e., $y_t = \begin{pmatrix} y_{1,t}^T & y_{2,t}^T \end{pmatrix}^T$. Note that the cases where either $y_{1,t}$ or $y_{2,t}$ are present are obviously special cases. First of all, the measurements are conditionally independent given the state, implying that

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

Now, inserting (17) into (16) gives

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

which inserted in (15) finally results in

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

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

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

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 for computational reasons want to use a model suitable for the RBPF/MPF, 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

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

D. Map Estimation and Map Management

A simple map consists of a collection of map point 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 a very simple map parameterization. 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

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

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

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

It has to be checked 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 bijective with respect to the map $m_{j,t}$ this can be used to directly initialize the position from the measurement $y_{2,t}$. However, this is typically not the case, 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 ideas on how to reduce it. In the present context multirate sensors (sensors providing measurements with different sampling times) are typically used. This can be exploited in a way similar to that in [19]. The idea is that, 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. This approach involves 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 it will provide the true position and orientation with very high accuracy. This enables systematic evaluation of algorithms. The sensor and its mounting to the industrial robot is illustrated in Figure. 1.

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 basic 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 orientation.

In order to put the model in the RBPF/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

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

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

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

$$p_t = p_t^e, \qquad v_t = v_t^e, \qquad a_t = a_t^e, \qquad (25a)$$

$$w_t = w_t^e, \qquad b_{\omega,t} = b_{\omega,t}^b, \qquad b_{a,t} = b_{a,t}^b. \qquad (25b)$$

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 (strap-down),

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

The quaternion estimates are normalized, to make sure that they still parameterizes an orientation. Further details regarding orientation and coordinate systems are given in Appendix A.

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

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

$$+ \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}}_{G_t^k} \underbrace{\begin{pmatrix} w_{1,t} \\ w_{2,t} \\ w_{3,t} \\ w_{4,t} \end{pmatrix}}_{w_t^k} \qquad (27a)$$

$$\underbrace{\begin{pmatrix} p_{t+1} \\ q_{t+1} \end{pmatrix}}_{x_{t+1}^p} = \underbrace{\begin{pmatrix} p_t \\ q_t \end{pmatrix}}_{f_t^p(x_t^p)} + \underbrace{\begin{pmatrix} TI & \frac{T^2}{2}I & 0_{3\times9} \\ 0_{4\times3} & 0_{4\times9} & -\frac{T}{2}\tilde{S}(q) \end{pmatrix}}_{A_t^p(x_t^p)} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x_t^k} + w_t^p,$$

$$(27b)$$

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

where

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

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

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 [24]–[26]. The inertial measurements are related to the states according

to,

$$y_{1,t} = \begin{pmatrix} y_{\omega,t} \\ y_{a,t} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 \\ -R(q_t)g^e \end{pmatrix}}_{h(x_t^p)}$$

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

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

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 [27], [28],

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

where

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

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 with the inertial measurements. This provides a good setup for fusing vision information with the inertial information. 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., [29].

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

$$y_{m_j,t} = h^c(m_{j,t}, p_t, q_t) + e_{3,t} \quad (32a)$$

$$\approx \underbrace{h_j^c(\hat{m}_{j,t|t-1}, p_t, q_t) - H_{j,t}\hat{m}_{j,t|t-1}}_{h(x_t^p)}$$

$$+ H_{j,t}m_{j,t} + e_{3,t}, \quad j = 1, \dots, M_t, \quad (32b)$$

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

$$H_{j,t} = \frac{\partial h^c}{\partial m_j} = \frac{\partial h^c}{\partial m_j^c} \frac{\partial m_j^c}{\partial m_j}, \quad (33)$$
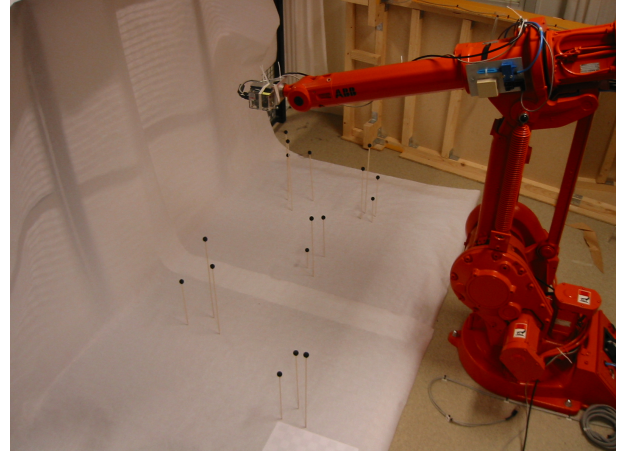

Figure 2. The SLAM setup in the industrial robot laboratory, with the camera and IMU sensor in the robot manipulator. The scene consists of black balls on a stick, making the feature extraction very simple.

The two partial derivatives in (33) are given by

$$\frac{\partial h^c}{\partial m_j^c} = \begin{pmatrix} \frac{1}{z^c} & 0 & -\frac{x^c}{(z^c)^2} \\ 0 & \frac{1}{z^c} & -\frac{y^c}{(z^c)^2} \end{pmatrix}, \quad (34a)$$

$$\frac{\partial m_j^c}{\partial m_j} = R(q_t^{cb})R(q_t^{be}). \quad (34b)$$

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 [30], which basically extracts well-defined corners in an image. There are more elaborate detectors available, for example the so-called *scale-invariant feature transform* (SIFT) [31]. 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. From a computer vision perspective the current environment is rather simple, hence fast and simple corner detectors can be successfully applied.

B. Experiment Setup

The scene surrounding the platform is equipped with easily detectable features placed in the vicinity as depicted in Figure. 2. In Table I the parameters in the SLAM example are presented.

In the example presented here, the noise level in the measurements is such that a simplified version of the NN method can be used to solve the data association problem. 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 suboptimally, assigning the closest one first. This seems to work perfectly well for the present application.

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 Figure. 3 the

Table I

SYSTEM, FILTER, AND SENSOR PARAMETERS FOR THE EXPERIMENT.

Process Noise	
Cov. Acceleration	$Q_a = diag\left(0.5^2 \quad 0.5^2 \quad 0.5^2\right)$
Cov. Bias angular rate	$Q_{b_\omega} = diag\left(10^{-10} \quad 10^{-10} \quad 10^{-10}\right)$
Cov. Bias acceleration	$Q_{b_a} = diag\left(0.001^2 \quad 0.001^2 \quad 0.001^2\right)$
Cov. Angular rate	$Q_\omega = diag\left(0.001^2 \quad 0.001^2 \quad 0.001^2\right)$
Measurement Noise	
Cov. Camera	$R_c = diag\left(0.01^2 \quad 0.01^2\right)$
Cov. Accelerometer	$R_a = diag\left(0.02^2 \quad 0.02^2 \quad 0.03^2\right)$
Cov. Gyroscope	$R_w = diag\left(0.02^2 \quad 0.03^2 \quad 0.03^2\right)$
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	$N = 100$

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

true trajectory is depicted. Note 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 Figure. 3 the result from dead-reckoning, i.e., direct double integration of the IMU acceleration data (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 provided from accurate measurements in the robot. 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 significantly improved 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 estimates. In the current experiment the bias terms were not used.

In Figure. 4 the measurement from the accelerometers, the

MPF-SLAM, 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 Figure. 5 for a specific time. Note that the uncertainty is visible in the spread of the particle cloud.

Figure 5. Vision measurements from the feature extraction (cross) and current map represented by the particle cloud (red dots). Please note that this figure has to be viewed in color.
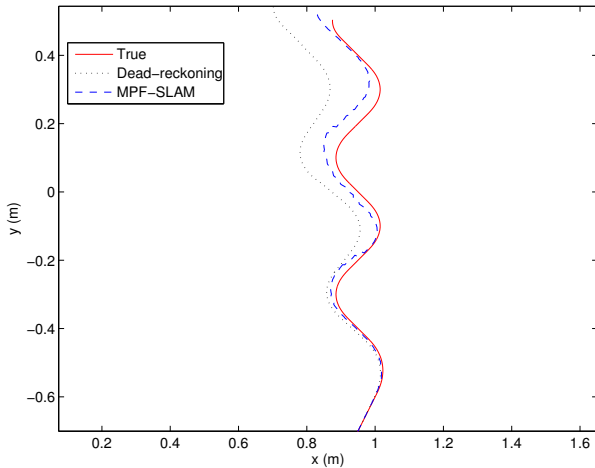

Figure 3. The ground truth position from the industrial robot, dead-reckoned position from inertial sensor data (5 seconds in 100 Hz), the estimate from the MPF-SLAM algorithm with vision measurements.
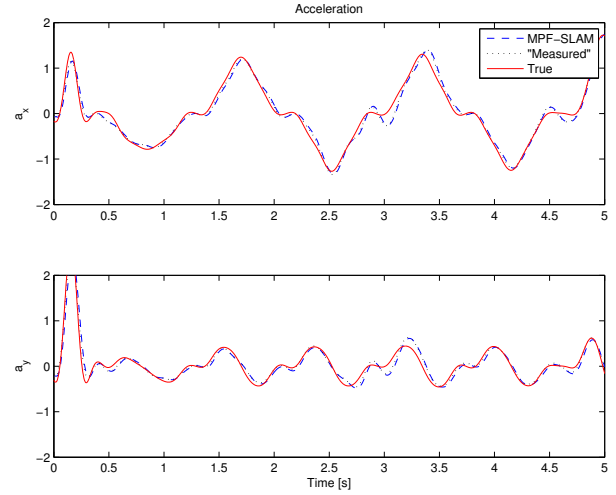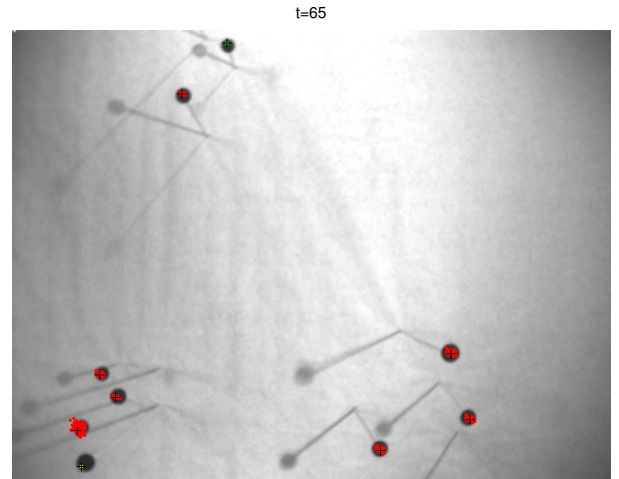
V. CONCLUSION

This contribution has introduced a Rao-Blackwellized/marginalized particle filtering framework for solving the SLAM problem, enabling high dimensional state vectors for the moving platform. The idea draws on the FastSLAM algorithm, but rather than using only the map, we also include all the states corresponding to a linear Gaussian sub-structure in the analytically solvable Kalman filter part. The approach was validated using a simple application

example, where inertial measurements were fused with vision measurements. The correctness of the result was assessed using the ground truth.

ACKNOWLEDGMENT

APPENDIX A
COORDINATE SYSTEMS

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

$$x_{\mathrm{B}} = R_{\mathrm{BA}} x_{\mathrm{A}}.$$

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

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

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

$$u_{\mathrm{B}} = \bar{q}_{\mathrm{A}} \odot u_{\mathrm{A}} \odot q_{\mathrm{A}},$$

where u_{A} is the quaternion extension of the vector x_A, i.e., $u_{\mathrm{A}} = \begin{pmatrix} 0 & x_{\mathrm{A}}^T \end{pmatrix}^T$ and $\odot$ represents quaternion multiplication. Furthermore, $\bar{u}$ denotes the quaternion conjugate. See e.g., [32], [33] for an introduction to unit quaternions and other 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, Jul. 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, Sep. 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, Jun. 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, Jul. 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, Jun. 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 *15th European Signal Processing Conference (EUSIPCO)*, Poznan', Poland, Sep. 2007, accepted for publication.

[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] 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.

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

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

[27] 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.

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

[29] 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.

[30] 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.

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

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

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