J Intell Robot Syst
DOI 10.1007/s10846-008-9301-y

Particle Filter SLAM with High Dimensional Vehicle Model

David Törnqvist · Thomas B. Schön · Rickard Karlsson · Fredrik Gustafsson

Received: 26 May 2008 / Accepted: 1 December 2008

Abstract This work presents a particle filter method closely related to FastSLAM for solving the simultaneous localization and mapping (SLAM) problem. Using the standard FastSLAM algorithm, only low-dimensional vehicle models can be handled due to computational constraints. In this work, an extra factorization of the problem is introduced that makes high-dimensional vehicle models computationally feasible. Results using experimental data from an unmanned aerial vehicle (helicopter) are presented. The proposed algorithm fuses measurements from on-board inertial sensors (accelerometer and gyro), barometer, and vision in order to solve the SLAM problem.

Keywords Rao-Blackwellized/marginalized particle filter · Sensor fusion · Simultaneous localization and mapping · Inertial sensors · UAV · Vision

1 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) [13, 18] has proved to be an enabling technology for many applications of this kind, in

D. Törnqvist (✉) · T. B. Schön · R. Karlsson · F. Gustafsson
Division of Automatic Control, Department of Electrical Engineering,
Linköping University, 581 83 Linköping, Sweden
e-mail: tornqvist@isy.liu.se

T. B. Schön
e-mail: schon@isy.liu.se

R. Karlsson
e-mail: rickard@isy.liu.se

F. Gustafsson
e-mail: fredrik@isy.liu.se

Published online: 21 January 2009

particular when the observations are complicated nonlinear functions of the position and heading [19]. Furthermore, the *Rao-Blackwellized particle filter* (RBPF), also denoted the *marginalized particle filter* (MPF) [1, 5, 7, 14, 15, 32], enables estimation of velocity, acceleration, and sensor error models by utilizing any linear Gaussian substructure in the model, which is fundamental for performance in applications as surveyed in [33]. As described in [32], the RBPF splits the state vector x_t into two parts, one part x_t^p, which is estimated using the PF, 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,

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

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, a PF is necessary.

Simultaneous localization and mapping (SLAM) is an extension of the localization or positioning problem to the case where the environment is unmodeled and has to be mapped online. An introduction to the SLAM problem is given in the survey papers [2, 16] and the recent book [38]. From a sensor point of view, there are two ways to tackle this problem. The first way is to use only one sensor, such as vision, see, e.g., [9–11, 17], and the second way is to fuse measurements from several sensors. This work considers the latter. The FastSLAM algorithm introduced in [26] has proved 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

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

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) [38]. This implies that all platform states must be estimated by the PF, which is computationally demanding for high-order models. Note that there have been successful applications with a higher state dimension, see [37], where a six-dimensional state vector was used. However, for yet higher state dimensions, relying purely on the PF will still be problematic.

Parallelling the evolution of PF applications to high dimensional state vectors, the aim of this contribution is to build on our earlier work [22, 34] which unify the ideas presented in [32] and [35]. This is done in order to extend the FastSLAM algorithm [26] to be able to cope with high dimensional state vectors as well. Basically, the main result follows from

$$p\left(x_{1:t}^p, x_t^k, m_t | y_{1:t}\right) = p\left(m_t | x_t^k, x_{1:t}^p, y_{1:t}\right) p\left(x_t^k | x_{1:t}^p, y_{1:t}\right) p\left(x_{1:t}^p | y_{1:t}\right). \tag{3}$$

The derived algorithm is applied to experimental data from an autonomous aerial vehicle using the RMAX helicopter platform (Fig. 1), although the framework is general and could be applied to other platforms as well. The main navigation sensor

Fig. 1 The Yamaha RMAX helicopter used in the experiments. The on-board system is equipped with an IMU sensor (accelerometer and gyro) and a vision sensor. The on-board GPS receiver is used for evaluation only

unit consists of three accelerometers, three gyros, a pressure sensor, and a camera. A global positioning system (GPS) is used only for evaluation purposes.

In Section 2, the problem under consideration is formulated in more detail. The proposed algorithm is given and explained in Section 3. This algorithm is then applied to solve the SLAM problem for a unmanned aerial vehicle (UAV) in Section 4. This section also provides a thorough explanation of the model used and how the vision measurements are computed. Finally, the conclusions are given in Section 5.

2 Problem Formulation

The aim of this work is to solve the SLAM problem when the state dimension of the platform (UAV) 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

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

where x_t^p denotes the states of the platform that are estimated by the PF, 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 (landmarks) m_t, are estimated using Kalman filters. The map states m_t consists of the entire map at time t, i.e.,

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

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 *probability density function* (PDF),

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

In other words, a nonlinear filtering problem is solved, providing an estimate of (6). The *key* factorization, which makes it possible to solve this problem successfully, is

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

where the trajectory $x_{1:t}^p$ is estimated to make the last factorization possible. It is also assumed that $m_{j,t}$ is independent of x_t^k, which will be seen in the model (8). In order to devise an estimator for (6), a *system model* and a *measurement model* are needed. The former describes the dynamic behavior 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. A general algorithm is sought, which is applicable to many different platforms (aircraft, helicopters, cars, etc.). Hence, the model structure should be as general as possible,

$$x_{t+1}^p = f_t^p\left(x_t^p\right) + A_t^p\left(x_t^p\right) x_t^k + G_t^p\left(x_t^p\right) w_t^p, \tag{8a}$$

$$x_{t+1}^k = f_t^k\left(x_t^p\right) + A_t^k\left(x_t^p\right) x_t^k + G_t^k\left(x_t^p\right) w_t^k, \tag{8b}$$

$$m_{j,t+1} = m_{j,t}, \tag{8c}$$

$$y_{1,t} = h_{1,t}\left(x_t^p\right) + C_{1,t}\left(x_t^p\right) x_t^k + e_{1,t}, \tag{8d}$$

$$y_{2,t}^{(j)} = h_{2,t}\left(x_t^p\right) + H_{j,t}\left(x_t^p\right) m_{j,t} + e_{2,t}^{(j)}, \tag{8e}$$

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

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

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

$$e_{1,t} \sim \mathcal{N}(0, R_{1,t}), \tag{8g}$$

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

Finally, x_0^k is Gaussian,

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

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 models quantities related to the platform, whereas the latter also involves the map states. Section 4 describes a detailed application example using experimental data, where (8d) is used to model inertial sensors and a pressure sensor and (8e) is used to model a camera.

3 Particle Filter for SLAM Utilizing Structure

This section is devoted to deriving and explaining the proposed SLAM algorithm on a rather detailed level. However, when we make use of standard results, we just provide the necessary references. The algorithm is given first and the steps are then discussed in more detail.

3.1 Algorithm

The algorithm presented in this paper draws on several rather well known algorithms. It is based on the RBPF/MPF method [1, 5, 7, 14, 15, 32]. The FastSLAM algorithm [26] is extended by not only including the map states in the linear part, but also the states corresponding to a linear Gaussian substructure present in the model for the platform. Assuming that the platform is modeled in the form (8), the SLAM algorithm utilizing structure is given in Algorithm 1.

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 PF can be straightforwardly incorporated into Algorithm 1. To give one example, the so-called FastSLAM 2.0 makes use of an improved proposal distribution in step 6b [27].

Theorem 1 *Using the model given in* (8)*, the conditional probability density functions for* x_t^k *and* x_{t+1}^k *are given by*

$$p\left(x_t^k | x_{1:t}^p, y_{1:t}\right) = \mathcal{N}\left(\hat{x}_{t|t}^k, P_{t|t}\right), \tag{10a}$$

$$p\left(x_{t+1}^k | x_{1:t+1}^p, y_{1:t}\right) = \mathcal{N}\left(\hat{x}_{t+1|t}^k, P_{t+1|t}\right), \tag{10b}$$

where

$$\hat{x}_{t|t}^k = \hat{x}_{t|t-1}^k + K_t\left(y_{1,t} - h_{1,t} - C_t \hat{x}_{t|t-1}^k\right), \tag{11a}$$

$$P_{t|t} = P_{t|t-1} - K_t S_{1,t} K_t^T, \tag{11b}$$

$$S_{1,t} = C_t P_{t|t-1} C_t^T + R_{1,t}, \tag{11c}$$

$$K_t = P_{t|t-1} C_t^T S_{1,t}^{-1}, \tag{11d}$$

and

$$\hat{x}_{t+1|t}^k = \bar{A}_t^k \hat{x}_{t|t}^k + G_t^k \left(Q_t^{kp}\right)^T \left(G_t^p Q_t^p\right)^{-1} z_t + f_t^k + L_t\left(z_t - A_t^p \hat{x}_{t|t}^k\right),$$

$$P_{t+1|t} = \bar{A}_t^k P_{t|t} \left(\bar{A}_t^k\right)^T + G_t^k \bar{Q}_t^k \left(G_t^k\right)^T - L_t S_{2,t} L_t^T, \tag{12a}$$

$$S_{2,t} = A_t^p P_{t|t} \left(A_t^p\right)^T + G_t^p Q_t^p \left(G_t^p\right)^T, \tag{12b}$$

$$L_t = \bar{A}_t^k P_{t|t} \left(A_t^p\right)^T S_{2,t}^{-1}, \tag{12c}$$

Algorithm 1 (PF for SLAM utilizing structure)

1. Initialize N particles according to

$$x_{1|0}^{p,(i)} \sim p\left(x_{1|0}^{p}\right),$$
$$x_{1|0}^{k,(i)} = \bar{x}_{1|0}^{k},$$
$$P_{1|0}^{k,(i)} = \bar{P}_{1|0}, \qquad i = 1, \ldots, N.$$

2. If there are new map-related measurements available, compute the necessary correspondences to the existing states, otherwise proceed to step 3.
3. Compute the importance weights according to

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

and normalize $\tilde{\gamma}_t^{(i)} = \gamma_t^{(i)} / \sum_{j=1}^{N} \gamma_t^{(j)}$.

4. For each $i = 1, \ldots, N$ draw a new particle $x_{t|t}^{(i)}$ with replacement (resample) according to,

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

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,

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

   where $\hat{x}_{t|t}^{k,(i)}$ and $P_{t|t}^{(i)}$ are given in (11).
   (b) Time update for the nonlinear state,

$$x_{t+1|t}^{p,(i)} \sim p\left(x_{t+1} | x_{1:t}^{p,(i)}, y_{1:t}\right).$$

   (c) Kalman filter time update,

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

   where $\hat{x}_{t+1|t}^{k,(i)}$ and $P_{t+1|t}^{(i)}$ are given by (12).
7. Set $t := t + 1$ and repeat from step 2.

where

$$z_t = x_{t+1}^{p} - f_t^{p}, \tag{13a}$$
$$\bar{A}_t^k = A_t^k - G_t^k \left(Q_t^{kp}\right)^T \left(G_t^p Q_t^p\right)^{-1} A_t^p, \tag{13b}$$
$$\bar{Q}_t^k = Q_t^k - \left(Q_t^{kp}\right)^T \left(Q_t^p\right)^{-1} Q_t^{kp}. \tag{13c}$$

Proof The proof is derived in [32] for the case without map features but with linear Gaussian dynamics as a substructure. The extension, including the linear Gaussian map substructure, falls within the same framework. □

3.2 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

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

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

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

where

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

Consider the case where both $y_{1,t}$ and $y_{2,t}$ are present, i.e., $y_t = \big(y_{1,t}^T \; y_{2,t}^T\big)^T$. 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

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

Now, inserting (17) into (16) gives

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

which, inserted in (15) finally results in

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

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

$$p\big(x_t^k|x_{1:t}^p, y_{1:t-1}\big) = \mathcal{N}\big(x_t^k|\hat{x}_{t|t-1}^k, P_{t|t-1}\big), \tag{20a}$$

$$p\big(m_{j,t}|x_{1:t}^p, y_{1:t-1}\big) = \mathcal{N}\big(m_{j,t}|\hat{m}_{j,t-1}, \Sigma_{j,t-1}\big), \tag{20b}$$

$$p\big(y_{1,t}|x_t^k, x_t^p\big) = \mathcal{N}\big(y_{1,t}|h_{1,t} + C_t x_t^k, R_1\big), \tag{20c}$$

$$p\big(y_{2,t}^{(j)}|x_t^p, m_{j,t}\big) = \mathcal{N}\big(y_{2,t}^{(j)}|h_{2,t} + H_{j,t}m_{j,t}, R_2^j\big). \tag{20d}$$

Here, it is important to note that the standard FastSLAM approximation (linearization) has been invoked in order to obtain (20d). That is, the measurement equation often has to be linearized with respect to the map states $m_{j,t}$ in order to comply with (8e). The reason for this approximation is that we want to use a model suitable for the RBPF/MPF; otherwise, the dimension will be much too large for the PF to handle. Using (20), the integrals in (19) can now be solved, resulting in

$$p\left(y_t|x_{1:t}^{p,(i)}, y_{1:t-1}\right) = \mathcal{N}\left(y_{1,t}|h_{1,t} + C_t\hat{x}_{t|t-1}^{k,(i)}, C_t P_{t|t-1}^{(i)} C_t^T + R_1\right) \times \prod_{j=1}^{M_t} \mathcal{N}\left(y_{2,t}^{(j)}|h_{2,t} + H_{j,t}\hat{m}_{j,t-1}, H_{j,t}\Sigma_{j,t-1}(H_{j,t})^T + R_2^j\right). \quad (21)$$

3.3 Map Estimation and Map Management

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

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

Note that this is a very simple map parametrization. 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

$$m_{j,t} = m_{j,t-1} + K_{j,t}\left(y_{2,t}^{(j)} - h_{2,t}\right), \quad (22a)$$

$$\Sigma_{j,t} = \left(I - K_{j,t}H_{j,t}^T\right)\Sigma_{j,t-1}, \quad (22b)$$

$$K_{j,t} = \Sigma_{j,t-1}H_{j,t}^T\left(H_{j,t}\Sigma_{j,t-1}H_{j,t}^T + R_2\right)^{-1}. \quad (22c)$$

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

$$m_{j,t} = m_{j,t-1}, \quad (23a)$$

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

Finally, initialization of new map entries has 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 we cannot uniquely initialize the position of the corresponding map entry. This can be handled in different ways. In Section 4 the vision sensor and different techniques are briefly discussed.

4 Application Example

In this section, we provide a description of the SLAM application, where Algorithm 1 is used to fuse measurements from a camera, three accelerometers, three gyros,

and an air-pressure sensor. The sensors are mounted on the RMAX helicopter shown previously in Fig. 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. The vision system can extract and track features that are used in SLAM to reduce the inertial drift and the bias in the inertial measurement unit (IMU) sensor.

The dynamic model used to explain the helicopter motion is introduced in Section 4.1, together with measurement models for the various sensors. Further details regarding the UAV platform are provided in Section 4.2. Finally, the experimental setup and the results are given in Section 4.3 and Section 4.4, respectively.

4.1 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, where the latter is used to parametrize the orientation of the UAV.

In order to relate the model to 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 PF x_t^p. In our case, x_t^p contains the orientation and position; the other states are contained in x_t^k. This will give a linear system for x_t^k given x_t^p, as will be seen in the model (27). Hence, define

$$x_t^k = \left(v_t^T \; a_t^T \; (b_{\omega,t})^T \; (b_{a,t})^T \; \omega_t^T\right)^T, \tag{24a}$$

$$x_t^p = \left(p_t^T \; q_t^T\right)^T, \tag{24b}$$

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 frame an entity is expressed in. Here, the notation is simplified by suppressing the coordinate frame, which means that

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

$$\omega_t = \omega_t^b, \qquad b_{\omega,t} = b_{\omega,t}^b, \qquad b_{a,t} = b_{a,t}^b. \tag{25b}$$

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 (strap-down),

$$q_t = q_t^{be} = \left(q_{0,t} \; q_{1,t} \; q_{2,t} \; q_{3,t}\right)^T. \tag{26}$$

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

4.1.1 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.

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

$$\underbrace{\begin{pmatrix} p_{t+1} \\ q_{t+1} \end{pmatrix}}_{x_{t+1}^p} = \underbrace{\begin{pmatrix} p_t \\ q_t \end{pmatrix}}_{f_t^p\left(x_t^p\right)} + \underbrace{\begin{pmatrix} TI & \frac{T^2}{2} I & 0_{3\times 9} \\ 0_{4\times 3} & 0_{4\times 9} & \frac{T}{2}\tilde{S}(q_t) \end{pmatrix}}_{A_t^p\left(x_t^p\right)} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x_t^k} + \begin{pmatrix} \frac{T^3}{6} I & 0 \\ 0_{4\times 3} & \frac{T}{2}\tilde{S}(q_t) \end{pmatrix} \begin{pmatrix} w_{1,t} \\ w_{2,t} \end{pmatrix}, \tag{27b}$$

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

where

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

T denotes the sampling time, I denotes the 3×3 identity 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 $Q_t^k = \text{diag}(Q_a, Q_{b_\omega}, Q_{b_a}, Q_\omega)$.

4.1.2 Measurement Model—Inertial and Air Pressure 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 magnetic disturbances from the environment, it is just the accelerometers and gyroscopes that are used for positioning. There is also a barometer available $y_{p,t}$, measuring the altitude via the air pressure. The measurements from these sensors are antialias-filtered and down-sampled to 20 Hz. The frequency used is a tradeoff between computational load and lost accuracy. This sampling rate has shown to give good accuracy at reasonable computational load. For further details on inertial

sensors, see, for instance, [6, 30, 31, 39]. The measurements are related to the states according to

$$y_{1,t} = \begin{pmatrix} y_{p,t} \\ y_{\omega,t} \\ y_{a,t} \end{pmatrix} = \underbrace{\begin{pmatrix} p_{3,t} \\ 0 \\ -R(q_t)g^e \end{pmatrix}}_{h\left(x_t^p\right)} + \underbrace{\begin{pmatrix} & 0_{1\times 15} & \\ 0 & 0 & I & 0\ I \\ 0 & R(q_t) & 0 & I\ 0 \end{pmatrix}}_{C\left(x_t^p\right)} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x_t^k} + \underbrace{\begin{pmatrix} e_{1,t} \\ e_{2,t} \\ e_{3,t} \end{pmatrix}}_{e_t},$$

which obviously is in the form (8). The measurement noise e_t is assumed Gaussian with covariance $R_t = \text{diag}(R_\omega, R_a)$ and the gravity vector $g^e = (0\,0\,-9.82)^T$.

4.1.3 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 unit focal length, according to [25] and [21],

$$y_{2,t} = y_{m_j,t} = \underbrace{\frac{1}{z_t^c}\begin{pmatrix} x_t^c \\ y_t^c \end{pmatrix}}_{h^c(m_{j,t},p_t,q_t)} + e_{3,t}, \tag{29}$$

where

$$m_{j,t}^c = \begin{pmatrix} x_t^c \\ y_t^c \\ z_t^c \end{pmatrix} = R\left(q_t^{cb}\right)R\left(q_t^{be}\right)(m_{j,t} - p_t) + r^c \tag{30}$$

is the position of map entry $m_{j,t}$ resolved in the camera frame. Here, r^c is a fixed vector representing the translation between the camera and the IMU (body), q_t^{cb} is the unit quaternion describing the rotation from the IMU to the camera, and $R(\cdot)$ is the rotation matrix in (38). The measurement noise is distributed as

$$e_{3,t} \sim \mathcal{N}(0, R_c). \tag{31}$$

The UAV is equipped with a camera, which is synchronized in time with the inertial measurements. Images are available at 4 Hz in a resolution of 384×288 pixels. In order to use vision for feature extraction and estimation, we have made use of standard camera calibration techniques; see, e.g., [41].

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

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

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

$$H_{j,t} = \frac{\partial h^c}{\partial m_j} = \frac{\partial h^c}{\partial m_j^c} \frac{\partial m_j^c}{\partial m_j}. \tag{33}$$

The two partial derivatives in (33) are given by

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

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

4.1.4 Obtaining Vision Measurements

In order to receive a camera measurement in the form (29), interest points or features have to be identified in the image. This is step 2 in Algorithm 1. In this application example, features are found using the Harris detector [20], which basically extracts well-defined corners in an image. These feature points are then searched for in the subsequent images using cross-correlations [25] according to Algorithm 2.

Algorithm 2 (Extracting and managing map entries)

1. Initialization. Search the entire image for features using the Harris detector. Save an 11-by-11 pixel patch around each corner.
2. Predict positions of features detected in previous images. Match saved patches in small search regions around the predicted feature positions using normalized cross-correlations. Also, apply a weighted criterion to the matching procedure so that a match close to the prediction is more likely.
3. Outlier rejection. If a matched feature is far from the predicted position compared to other features, the measurement is discarded.
4. In areas of the image without features, search for new features using the Harris detector. Around each detected corner, an 11-by-11 pixel patch is extracted and stored.
5. Initialize the detected features in the map.

The feature detection in Algorithm 2 is performed in the 2-D image plane. However, the features found have to be initialized into the filter 3-D map. In this application, the features are known to be close to the ground, and we have a good estimate of the altitude thanks to the air pressure sensor. The features are therefore initialized on the estimated ground level, and adjustments are made by implicit triangulation in the PF. In a more general case, where the depths of the features are unknown, there are several methods available for intialization. For example, the initialization can be delayed a few time steps until the feature has been seen from several angles and its depth can be estimated by triangulation. Another alternative is to use an inverse depth parametrization for some time as in [8].

This algorithm has been shown to work reliably on our flight data. However, improvements can be achieved in both computational speed and detection reliability.

There are more elaborate detectors available, for example, the *scale-invariant feature transform* (SIFT) [24], the *speeded up robust features* (SURF) [4], or the fast corner detector [28, 29]. It is also possible to refine the feature detection process even further by estimating the orientation of the patches [11]. From a computer vision perspective, the current environment is rather simple; hence, fast and simple corner detectors can be successfully applied.

4.2 UAV Platform

The algorithm proposed has been tested using flight-test data collected from an autonomous helicopter (UAV) developed during the WITAS Project [12]. The UAV is based on a commercial Yamaha RMAX helicopter (Fig. 1). The total helicopter length is 3.6 m (including main rotor); it is powered by a 21-hp, two-stroke engine; and it has a maximum take-off weight of 95 kg.

The avionics developed during the WITAS Project is integrated with the RMAX platform, and it is based on three computers and a number of sensors. The platform developed is capable of fully autonomous flight from take-off to landing.

The sensors used for the navigation algorithm described in this paper consist of an IMU (three accelerometers and three gyros), which provides the helicopter's acceleration and angular rate along the three body axes; a barometric altitude sensor; and a monocular CCD video camera. GPS position information is not used in the navigation filter described here.

The primary flight computer is a PC104 Pentium III, 700 MHz. It implements the low-level control system, which includes the control modes (take-off, hovering, path following, landing, etc.), sensor data acquisition, and communication with the helicopter platform. The second computer is also a PC104 Pentium III, 700 MHz, handling the image processing functionalities and controlling the camera pan-tilt unit. The third computer is a PC104 Pentium-M, 1.4 GHz, taking care of high-level functionalities such as path-planning, task-planning, etc.

4.3 Experiment Setup

The flight data were collected during a flight-test campaign in a training area in the south of Sweden and processed off-line. The helicopter autonomously flew a preplanned path using a path following functionality implemented in the software architecture [40]. The helicopter altitude was 60 m above the ground and the flight speed was 3 m/s. The video camera was looking downwards and fixed to the helicopter body. The video was recorded on-board and synchronized with the sensor data. The synchronization is performed by automatically turning on a light diode when the sensor data starts to be recorded. The light diode is visible in the camera image. The video is recorded on tape using an on-board video recorder and the synchronization with the sensor data is done manually off-line. This procedure allows for synchronization accuracy of about 40 ms. The video sequence is recorded at a 25-Hz frame rate. For the experiment described here, the video frames were sampled at 4 Hz. The on-board sensor data are recorded at different sample rates, although all sensors but the video are resampled to 20 Hz. Table 1 provides the characteristics of the sensors used in the experiment. The PF uses 200 particles.

Table 1 Available characteristics of the sensor used in the navigation algorithm

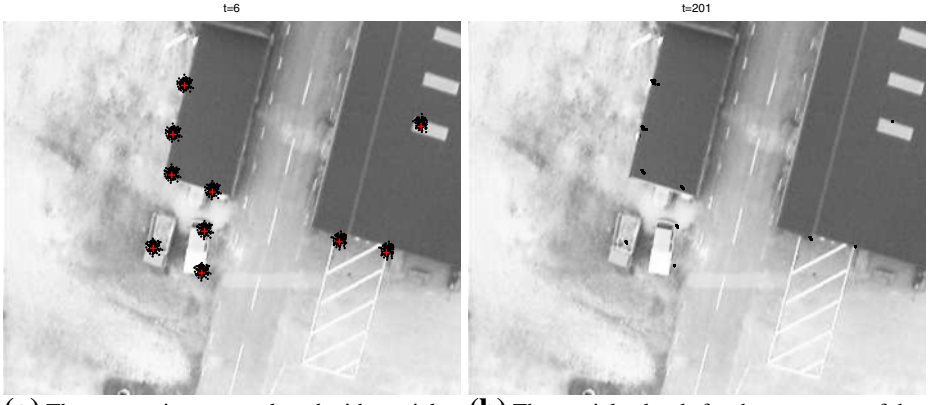
Sensor	Output rate	Resolution	Bias
Accelerometers	66 Hz	1 mG	13 mG
Gyros	200 Hz	0.1°/s	< 0.1°/s
Barometer	40 Hz	0.1 m	–
Vision	4 Hz	384 × 288 pixels	–

4.4 Experimental Results

In Fig. 2a, the landmarks/map are depicted using the particle clouds overlayed on the current camera image. Furthermore, Fig. 3a shows the Cartesian 2D position for the RBPFSLAM method and compares it against a GPS-based solution. Since the SLAM method, without closing the loop, is a dead-reckoning solution, it is expected to have some drift. Here, the drift is greatly reduced, compared to dead-reckoning using the inertial sensors alone. In Fig. 3b, the altitude (relative to the starting height) is depicted for the SLAM method and compared against the GPS-based reference and measured air pressure.

Estimating the altitude using only vision and inertial data is problematic since the vision measurement model will not get sufficient information in this direction. This is a well-known drawback of a monocamera solution. Hence, in order to reduce or remove a substantial drift in altitude, a pressure sensor is used.

Another important thing to note with the PFimplementation of SLAM is the degeneration of the map over time. This problem has previously been discussed in the literature, see, e.g., [3]. The resampling causes the map associated with the most probable features to be copied and the others to be discarded. For mapped features that have been out of sight for a while, the map will be the same for all particles after some time. Figure 2b shows the particles representing the map roughly 10 s after that area is left. The degeneration of the particles is almost complete and most particles

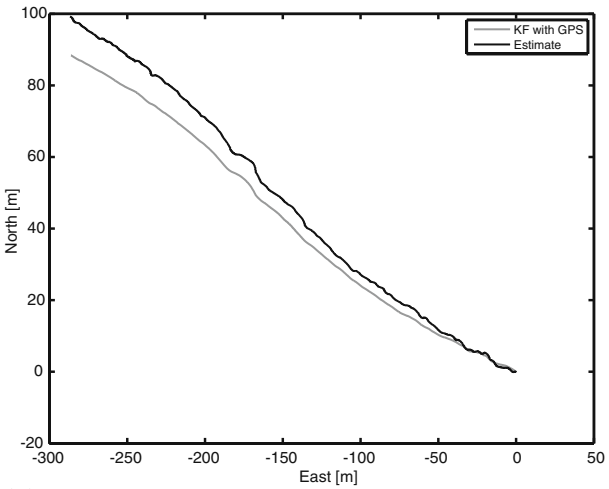


(a) The camera image overlayed with particle clouds for the map at time $t = 6$.

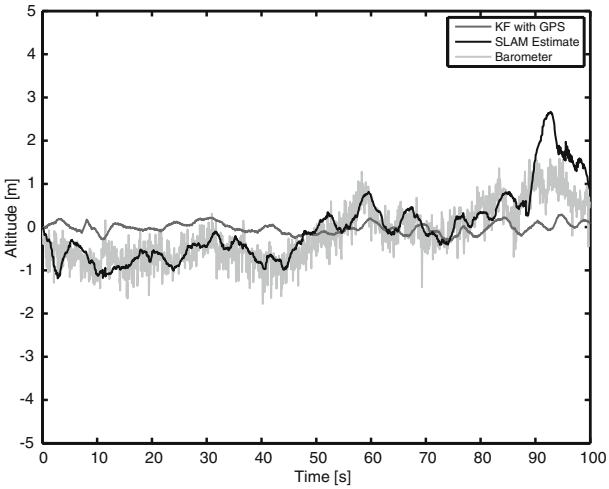
(b) The particle clouds for the same part of the map at time $t = 201$, overlayed on the image from $t = 6$. (10 seconds layer than in (a))

Fig. 2 The scenario seen from the on-board vision sensor, together with particle clouds representing the landmarks/map features and *crosses* showing the measured landmark positions

Fig. 3 Position and altitude of the RMAX helicopter



(a) 2D position from the SLAM method and the GPS based reference.



(b) Altitude (relative to the starting height) for SLAM, GPS refer–rence and measured with barometer.

are in the same point. This is not a problem in our example, but would be in the case of a loop-closure. The cross-correlation between the mapped features would, in such an event, correct even out-of-sight features. This capability would be limited here since the information about the cross-correlation among the features lies in the diversity of maps in the particles.

5 Conclusion

In this paper, a PF solution to the SLAM problem, capable of handling high-dimensional vehicle models, is presented. The solution includes an extra factorization

of the filtering density compared to the standard FastSLAM algorithm. The FastSLAM algorithm factorizes the filtering distribution into two parts. Then, a PF is used for filtering the states of the vehicle platform, and a Kalman filter bank handles the landmarks (map). In the approach presented here, a linear Gaussian substructure in the vehicle dynamics is handled in the Kalman filter bank as well. Thus, lowering the dimension of the PF, which decreases the computational load.

The UAV application consists of an RMAX helicopter, equipped with an IMU sensor (accelerometer and gyro), a pressure sensor, and a vision sensor. An on-board GPS sensor is used for evaluation, i.e., the solution presented in this paper does not rely on GPS. In an experiment, the proposed sensor fusion PF-based SLAM algorithm was successfully evaluated. Without the SLAM method, the poor IMU performance would not be sufficient for navigation, whereas the SLAM technique reduces the drift introduced by the dead-reckoning sensor.

Acknowledgements The authors would like to thank Gianpaolo Conte for providing experimental data and the reviewers for very insightful comments. This work was supported by the strategic research center MOVIII and funded by the Swedish Foundation for Strategic Research, SSF.

Appendix

Coordinate Systems

Three coordinate frames are used in this paper. An earth-fixed (denoted with e) body or inertial sensor system (denoted with b) and a camera system (denoted c).

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

$$x_{\mathrm{B}} = R_{\mathrm{BA}} x_{\mathrm{A}}. \tag{35}$$

where R_{BA} is used to denote the rotation matrix describing the rotation from A to B. Hence, the rotation from A to C via B can be described as

$$R_{\mathrm{CA}} = R_{\mathrm{CB}} R_{\mathrm{BA}}. \tag{36}$$

Another way to represent rotations is using a unit quaternion q_{A}. In this framework, a vector can be rotated from A to B via

$$u_{\mathrm{B}} = \bar{q}_{\mathrm{A}} \odot u_{\mathrm{A}} \odot q_{\mathrm{A}}, \tag{37}$$

where u_{A} is the quaternion extension of the vector x_A, i.e., $u_{\mathrm{A}} = (0 \;\; x_{\mathrm{A}}^T)^T$, and $\odot$ denotes quaternion multiplication. Furthermore, $\bar{u}$ denotes the quaternion conjugate. See, e.g., [36] and [23] for an introduction to unit quaternions and other rotation parameterizations.

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

$$R(q) = \begin{pmatrix} \left(q_0^2 + q_1^2 - q_2^2 - q_3^2\right) & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & \left(q_0^2 - q_1^2 + q_2^2 - q_3^2\right) & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & \left(q_0^2 - q_1^2 - q_2^2 + q_3^2\right) \end{pmatrix}, \quad (38)$$

where $q = (q_0\ q_1\ q_2\ q_3)^T$.

References

1. Andrieu, C., Doucet, A.: Particle filtering for partially observed Gaussian state space models. J. R. Stat. Soc. **64**(4), 827–836 (2002)
2. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM): Part II. IEEE Robot. Autom. Mag. **13**(3), 108–117 (2006)
3. Bailey, T., Nieto, J., Nebot, E.: Consistency of the FastSLAM algorithm. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, 15–19 May 2006
4. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: speeded up robust features. In: Proceedings of the ninth European Conference on Computer Vision, Graz, 7–13 May 2006
5. Casella, G., Robert, C.P.: Rao-Blackwellisation of sampling schemes. Biometrika **83**(1), 81–94 (1996)
6. Chatfield, A.: Fundamentals of High Accuracy Inertial Navigation, vol. 174, 3rd edn. American Institute of Aeronautics and Astronautics, New York (1997)
7. Chen, R., Liu, J.S.: Mixture Kalman filters. J. R. Stat. Soc. **62**(3), 493–508 (2000)
8. Civera, J., Davison, A.J., Montiel, J.M.M.: Inverse depth parameterization for monocular SLAM. IEEE Trans. Robot. **24**, 932–945 (2008)
9. Davison, A.J.: Real-time simultaneous localisation and mapping with a single camera. In: Proceedings Ninth IEEE International Conference on Computer Vision, vol. 2, pp. 1403–1410. IEEE, Nice (2003)
10. Davison, A.J., Cid, Y.G., Kita, N.: Real-time 3D SLAM with wide-angle vision. In: Proceedings of the 5th IFAC/EUCON Symposium on Intelligent Autonomus Vehicles, Lisaboa, July 2004
11. Davison, A.J., Reid, I., Molton, N., Strasse, O.: MonoSLAM: real-time single camera SLAM. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1052–1067 (2007)
12. Doherty, P., Haslum, P., Heintz, F., Merz, T., Persson, T., Wingman, B.: A distributed architecture for intelligent unmanned aerial vehicle experimentation. In: Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems, Toulouse, 23–25 June 2004
13. Doucet, A., de Freitas, N., Gordon, N. (eds.): Sequential Monte Carlo Methods in Practice. Springer, New York (2001)
14. Doucet, A., Gordon, N., Krishnamurthy, V.: Particle filters for state estimation of jump Markov linear systems. IEEE Trans. Signal Process. **49**(3), 613–624 (2001)
15. Doucet, A., Godsill, S.J., Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering. Stat. Comput. **10**(3), 197–208 (2000)
16. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping (SLAM): Part I. IEEE Robot. Autom. Mag. **13**(2), 99–110 (2006)
17. Eade, E., Drummond, T.: Scalable monocular SLAM. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 469–476. IEEE, New York (2006)
18. Gordon, N.J., Salmond, D.J., Smith, A.F.M.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In: IEEE Proceedings on Radar and Signal Processing, vol. 140, pp. 107–113. IEEE, Piscataway (1993)
19. Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., Nordlund, P.-J.: Particle filters for positioning, navigation and tracking. IEEE Trans. Signal Process. **50**(2):425–437 (2002)
20. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the 4th Alvey Vision Conference, pp. 147–151, Manchester, August 1988

21. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2003)
22. Karlsson, R., Schön, T.B., Törnqvist, D., Conte, G., Gustafsson, F.: Utilizing model structure for efficient simultaneous localization and mapping for a UAV application. In: Proceedings of IEEE Aerospace Conference, Big Sky, 1–8 March 2008
23. Kuipers, J.B.: Quaternions and Rotation Sequences. Princeton Univerity Press, Princeton (1999)
24. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
25. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An invitation to 3-D vision—from images to geometric models. Interdisciplinary Applied Mathematics. Springer, New York (2006)
26. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM a factored solution to the simultaneous localization and mapping problem. In: Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, 28 July–1 August 2002
27. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1151–1156, Acapulco, 9–15 August 2003
28. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: Proceedings of IEEE International Conference on Computer Vision, Beijing, 17–20 October 2005
29. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Proceedings of European Conference on Computer Vision, Graz, May 2006
30. Savage, P.G.: Strapdown inertial navigation integration algorithm design part 1: attitude algorithms. J. Guid. Control Dyn. **21**(1), 19–28 (1998)
31. Savage, P.G.: Strapdown inertial navigation integration algorithm design part 2: velocity and position algorithms. J. Guid. Control Dyn. **21**(2), 208–221 (1998)
32. Schön, T., Gustafsson, F., Nordlund, P.-J.: Marginalized particle filters for mixed linear/nonlinear state-space models. IEEE Trans. Signal Process. **53**(7), 2279–2289 (2005)
33. Schön, T.B., Karlsson, R., Gustafsson, F.: The marginalized particle filter in practice. In: Proceedings of IEEE Aerospace Conference, Big Sky, March 2006
34. Schön, T., Karlsson, R., Törnqvist, D., Gustafsson, F.: A framework for simultaneous localization and mapping utilizing model structure. In: The 10th International Conference on Information Fusion, Quebec, August 2007
35. Schön, T., Törnqvist, D., Gustafsson, F.: Fast particle filters for multi-rate sensors. In: The 15th European Signal Processing Conference (EUSIPCO 2007), Poznan, September 2007
36. Shuster, M.D.: A survey of attitude representations. J. Astronaut. Sci. **41**(4), 439–517 (1993)
37. Sim, R., Elinas, P., Griffin, A., Shyr, M., Little, J.J.: Design and analysis of a framework for real-time vision-based SLAM using rao-blackwellised particle filters. In: Proceedings of the 3rd Canadian Conference on Computer and Robotic Vision, Quebec City, 7–9 June 2006
38. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT, Cambridge (2005)
39. Titterton, D.H., Weston, J.L.: Strapdown inertial navigation technology. IEE radar, sonar, navigation and avionics series. Peter Peregrinus, Stevenage (1997)
40. Wzorek, M., Conte, G., Rudol, P., Merz, T., Duranti, S., Doherty, P.: From motion planning to control—a navigation framework for an autonomous unmanned aerial vehicle. In: 21th Bristol UAV Systems Conference, Bristol, April 2006
41. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000)