Utilizing Model Structure for Efficient Simultaneous Localization and Mapping for a UAV Application

Rickard Karlsson[†], Thomas B. Schön[†], David Törnqvist[†], Gianpaolo Conte[‡], and Fredrik Gustafsson[†]

[†]Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping, Sweden
E-mail: {rickard, schon, tornqvist, fredrik}@isy.liu.se
Phone: +46 13-281000

[‡]Division of Artificial Intelligence and Integrated Computer System
Department of Computer and Information Science
Linköping University
SE–581 83 Linköping, Sweden
E-mail: giaco@ida.liu.se
Phone: +46 13-281000

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. Using the standard FastSLAM algorithm, only low-dimensional vehicle models are computationally feasible. In this work, an algorithm is introduced which merges FastSLAM and MPF, and the result is an algorithm for SLAM applications, where state vectors of higher dimensions can be used. Results using experimental data from a UAV (helicopter) are presented. The algorithm fuses measurements from on-board inertial sensors (accelerometer and gyro) and vision in order to solve the SLAM problem, i.e., enable navigation over a long period of time.

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), [1, 2], has proved 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 RBPF 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.

Figure 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.

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 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,

IEEEAC Paper #1103
1-4244-1488-1/08/$25.00 ©2008 IEEE

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

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

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.

Paralleling the evolution of PF applications to high dimensional state vectors, the aim of this contribution is to build on our earlier work [19] which unify the ideas presented in [9, 20]. This is done 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

$$\begin{aligned} p(x^p_{1:t},&x^k_t, m_t|y_{1:t}) \\ &= p(m_t|x^k_t, x^p_{1:t}, y_{1:t})p(x^k_t|x^p_{1:t}, y_{1:t})p(x^p_{1:t}|y_{1:t}). \end{aligned} \quad (3)$$

The derived algorithm is applied to experimental data from an autonomous aerial vehicle using the RMAX helicopter platform (Figure 1). The main navigation sensor unit, consists of three accelerometers, three gyros, a pressure sensor, and a camera. 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 an application example in Section 4. 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

$$x_t = \begin{pmatrix} (x^p_t)^T & (x^k_t)^T & m_t^T \end{pmatrix}^T, \quad (4)$$

where x^p_t denotes the states of the platform that are estimated by the particle filter, and x^k_t denotes the states of the platform that are linear-Gaussian given information about x^p_t. 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.,

$$m_t = \begin{pmatrix} m^T_{1,t} & \dots & m^T_{M_t,t} \end{pmatrix}^T, \quad (5)$$

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),

$$p(x^p_t, x^k_t, m_t|y_{1:t}). \quad (6)$$

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

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

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. 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,

$$x^p_{t+1} = f^p_t(x^p_t) + A^p_t(x^p_t)x^k_t + G^p_t(x^p_t)w^p_t, \quad (8a)$$
$$x^k_{t+1} = f^k_t(x^p_t) + A^k_t(x^p_t)x^k_t + G^k_t(x^p_t)w^k_t, \quad (8b)$$
$$m_{j,t+1} = m_{j,t}, \quad (8c)$$
$$y_{1,t} = h_{1,t}(x^p_t) + C_t(x^p_t)x^k_t + e_{1,t}, \quad (8d)$$
$$y^{(j)}_{2,t} = h_{2,t}(x^p_t) + H_{j,t}(x^p_t)m_{j,t} + e^{(j)}_{2,t}, \quad (8e)$$

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

$$w_t = \begin{pmatrix} w^p_t \\ w^k_t \end{pmatrix} \sim \mathcal{N}(0, Q_t), \quad Q_t = \begin{pmatrix} Q^p_t & Q^{pk}_t \\ (Q^{pk}_t)^T & Q^k_t \end{pmatrix}. \quad (8f)$$

To simplify the notation in the rest of the paper, denote $f^p_t(x^p_t)$ with f^p_t, $A^p_t(x^p_t)$ withA^p_t and so on. The measurement noise is assumed white and Gaussian distributed according to

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

Finally, x^k_0 is Gaussian,

$$x^k_0 \sim \mathcal{N}(\bar{x}_0, \bar{P}_0), \quad (8i)$$

and the density for x^p_0 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 4 describes a detailed application example using experimental data, where (8d) is used to model inertial sensors and (8e) is used to model a camera.

3. PARTICLE FILTER FOR SLAM UTILIZING STRUCTURE

This section is devoted to explaining the proposed SLAM algorithm on a rather detailed level. However, whenever we make use of standard results we just provide the necessary references.

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

$$\begin{aligned} x^{p,(i)}_{1|0} &\sim p(x^p_{1|0}), \\ x^{k,(i)}_{1|0} &= \bar{x}^k_{1|0}, \\ P^{k,(i)}_{1|0} &= \bar{P}_{1|0}, \qquad i = 1, \ldots, N, \end{aligned}$$

where N denotes the number of particles.

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

$$\gamma^{(i)}_t = p(y_t | x^{p,(i)}_{1:t}, y_{1:t-1}), \qquad i = 1, \ldots, N,$$

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

4. Draw N new particles with replacement (resampling) according to, for each $i = 1, \ldots, N$

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

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,

$$p(x^k_t | x^p_{1:t}, y_{1:t}) = \mathcal{N}(x^k_t | \hat{x}^{k,(i)}_{t|t}, P^{(i)}_{t|t}),$$

where $\hat{x}^{k,(i)}_{t|t}$ and $P^{(i)}_{t|t}$ are given in (11).

(b) Time update for the nonlinear particles,

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

(c) Kalman filter time update,

$$\begin{aligned} &p(x^k_{t+1} | x^p_{1:t+1}, y_{1:t}) \\ &\qquad = \mathcal{N}(x^k_{t+1|t} | \hat{x}^{k,(i)}_{t+1|t}, P^{(i)}_{t+1|t}), \end{aligned}$$

where $\hat{x}^{k,(i)}_{t+1|t}$ and $P^{(i)}_{t+1|t}$ 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 [21].

Theorem 1: Using the model given by (8), the conditional probability density functions for x^k_t and x^k_{t+1} are given by

$$p(x^k_t | x^p_{1:t}, y_{1:t}) = \mathcal{N}(\hat{x}^k_{t|t}, P_{t|t}), \tag{10a}$$

$$p(x^k_{t+1} | x^p_{1:t+1}, y_{1:t}) = \mathcal{N}(\hat{x}^k_{t+1|t}, P_{t+1|t}), \tag{10b}$$

where

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

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

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

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

and

$$\begin{aligned} \hat{x}^k_{t+1|t} &= \bar{A}^k_t \hat{x}^k_{t|t} + G^k_t (Q^{kp}_t)^T (G^p_t Q^p_t)^{-1} z_t \\ &\quad + f^k_t + L_t(z_t - A^p_t \hat{x}^k_{t|t}), \end{aligned} \tag{12a}$$

$$P_{t+1|t} = \bar{A}^k_t P_{t|t} (\bar{A}^k_t)^T + G^k_t \bar{Q}^k_t (G^k_t)^T - L_t S_{2,t} L_t^T, \tag{12b}$$

$$S_{2,t} = A^p_t P_{t|t} (A^p_t)^T + G^p_t Q^p_t (G^p_t)^T, \tag{12c}$$

$$L_t = \bar{A}^k_t P_{t|t} (A^p_t)^T S^{-1}_{2,t}, \tag{12d}$$

where

$$z_t = x^p_{t+1} - f^p_t, \tag{13a}$$

$$\bar{A}^k_t = A^k_t - G^k_t (Q^{kp}_t)^T (G^p_t Q^p_t)^{-1} A^p_t, \tag{13b}$$

$$\bar{Q}^k_t = Q^k_t - (Q^{kp}_t)^T (Q^p_t)^{-1} Q^{kp}_t. \tag{13c}$$

Proof: The derivation was done in [9] for the case without map features, but with linear Gaussian dynamics as a sub-structure. However, the extension by including the linear Gaussian map sub-structure falls within the same framework. ■

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

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

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

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

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

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

Now, inserting (17) into (16) gives

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

which inserted in (15) finally results in

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

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

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

Here it is important to note that the standard FastSLAM approximation has been invoked in order to obtain (20d). That is, the measurement equation often has to be linearized with respect to the map states $m_{j,t}$ in order to be written as (8e). 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

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

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 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

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

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

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

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 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 to the RMAX helicopter.

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 tracks features that are used in SLAM to reduce the inertial drift and bias in the IMU sensor.

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 parametrize 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

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

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

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

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

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

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),

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

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

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.

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

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

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

where

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

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 $Q_t^k = \mathrm{diag}(Q_a, Q_{b_\omega}, Q_{b_a}, Q_\omega)$.

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 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 anti-alias filtered and down-sampled to 20 Hz. For further details on inertial sensors, see for instance [22–24]. The measurements are related

to the states according to,

$$y_{1,t} = \begin{pmatrix} y_{p,t} \\ y_{\omega,t} \\ y_{a,t} \end{pmatrix} = \underbrace{\begin{pmatrix} p_{3,t} \\ 0 \\ -R(q_t)g^e \end{pmatrix}}_{h(x_t^p)} + \underbrace{\begin{pmatrix} & & 0_{1\times 15} & & \\ 0 & 0 & I & 0 & I \\ 0 & R(q_t) & 0 & I & 0 \end{pmatrix}}_{C(x_t^p)} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x_t^k} + \underbrace{\begin{pmatrix} e_{1,t} \\ e_{2,t} \\ e_{3,t} \end{pmatrix}}_{e_t}, \quad (29)$$

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

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 [25, 26],

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

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 (body) 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 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., [27].

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 [13] is in this case simply to linearize the camera measurement equations according to,

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

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

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

The two partial derivatives in (33) are given by

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

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

"Computing Vision Measurements"—In order to receive a camera measurement on the form (30), interest points or features has to be identified in the image. This is step 2 in Algorithm 1. In this application example, features are found using the Harris detector [28], which basically extracts well-defined corners in an image. These feature points are then searched for in the next images according to Algorithm 2.

Algorithm 2: Vision Algorithm

1. Initialization. Search the whole image for features with the Harris detector. Save an 11-by-11 pixel patch around each corner.

2. Predict positions of features detected in old images. Match saved patches in small search regions around the predicted positions. 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 with 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 done in the 2-D image plane. However, found features 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 adjustment are made by implicit triangulization in the particle filter. In a more general case, where the depth 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 achieved by triangulization. Another alternative is to use an inverse depth parametrization for some time as in [29]. Also, using a Kalman filter on information form could solve the problem of representing no information (infinte covariance) about the feature depth.

This algorithm has 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) [30], the *speeded up robust features* (SURF) [31] or the fast corner detector [32, 33]. It is also possible to refine the feature detection process even further by estimating the slope of an image plane [14]. From a computer vision perspective the current environment is rather simple, hence fast and simple corner detectors can be successfully applied.

UAV Platform

The algorithm proposed has been tested using flight-test data collected from an autonomous UAV helicopter developed during the WITAS Project [34]. The helicopter is based on a commercial Yamaha RMAX UAV helicopter (Figure. 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 inertial measurement unit (three accelerometers and three gyros) which provides helicopter's acceleration and angular rate along the three body axes, a barometric altitude sensor and a monocular CCD video camera mounted on a pan/tilt unit. GPS position information is not used in the navigation filter described here.

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

Experiment Setup

The flight data were collected during a flight-test campaign in a training area in south of Sweden. Sensor data and on-board video were recorded during an autonomous flight. The helicopter flew autonomously a pre-planned path using a path following functionality implemented in the software architecture [35]. The helicopter altitude was 60 m above the ground and the flight speed 3 m/s. The video camera was looking downwards and fixed with 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 start to be recorded. The light diode is visible in the camera frame. 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 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 rate. Table 1 provides the characteristics of the sensors used in the experiment.

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	$0.1^\circ/s$	$< 0.1^\circ/s$
Barometer	40 Hz	0.1 m	-
Vision	4 Hz	384x288 pixels	-

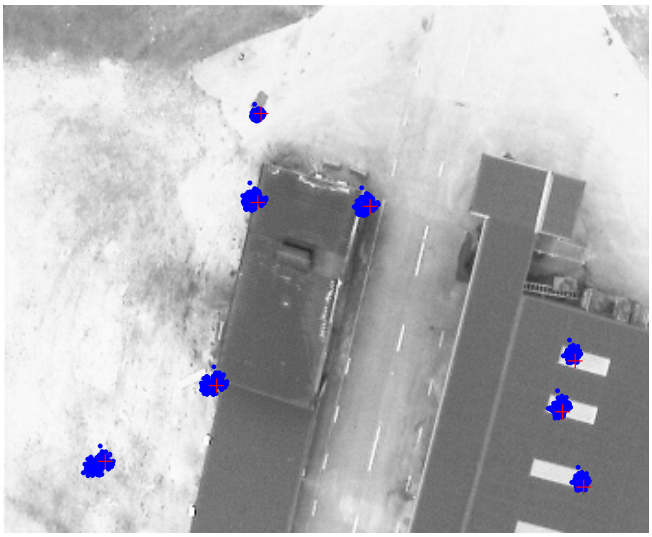


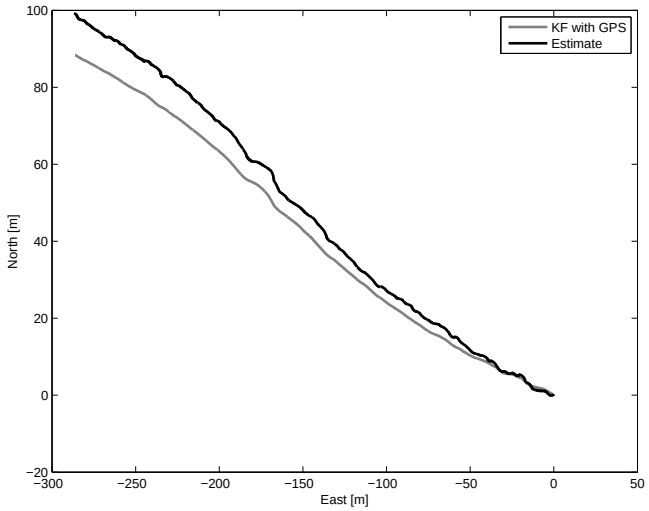
Figure 2. The scenario seen from the on-board vision sensor, together with particle clouds representing the landmarks/map features.

Experimental Results

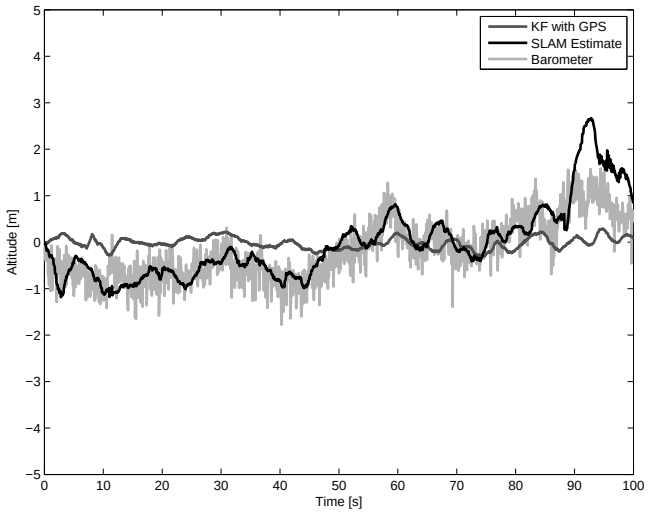
In Figure 2 the landmarks/map are depicted using the particle clouds on an image taken from the vision sensor.

In Figure 3 (a) the Cartesian 2D position is depicted for the RBPF SLAM method, and compared 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 has been greatly reduced if compared to dead-reckoning of the inertial sensors alone. In Figure 3 (b) the altitude (relative to the starting height) is depicted for the SLAM method and compared against the GPS based reference and measured air pressure.

The estimation of the altitude using only vision and IMU is problematic, since the vision measurement model will not get sufficient amount of information in this direction. Hence, in order to reduce or remove a substantial drift in altitude a pressure sensor is used.



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



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

Figure 3. Position and altitude of the RMAX helicopter.

Another thing to note with the particle filter implementation of SLAM is the degeneration of the map over time. 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 then be the same for all particles after some time. 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 cross correlation among features lies in the diversity of maps among the particles. This has been noted in, e.g., [36].

5. CONCLUSION

In this paper a FastSLAM algorithm incorporating a UAV platform with many state variables is presented. Traditionally, FastSLAM factorizes the problem in such a way that a particle filter solves the dynamics of the own platform and a Kalman filter bank handles the landmarks (map). Here, this is extended to also include linear Gaussian substructure in the state dynamics.

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 only. In an experiment the proposed sensor fusion particle filter based SLAM algorithm was successfully evaluated. Without the SLAM method the poor IMU performance is not sufficient for navigation, whereas the SLAM technique reduces the drift introduced by the dead-reckoning sensor.

ACKNOWLEDGMENT

The authors would like to thank MOVIII (Modeling, Visualization and Information Integration) and the Swedish research council (VR) for funding this work.

APPENDICES

A. COORDINATE SYSTEMS

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

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

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

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

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

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

where u_{A} is the quaternion extension of the vector x_A, i.e., $u_{\text{A}} = (0 \quad x_{\text{A}}^T)^T$ and $\odot$ represents quaternion multiplication. Furthermore, $\bar{u}$ denotes the quaternion conjugate. See e.g., [37, 38] 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. Schön, R. Karlsson, D. Törnqvist, and F. Gustafsson, "A framework for simultaneous localization and mapping utilizing model structure," in *Proceedings of the International Conference on Sensor Fusion*, 2007.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

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

[27] 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.

[28] 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.

[29] J. Civera, A. Davison, and J. Montiel, "Inverse depth to depth conversion for monocular slam," in *IEEE International Conference on Robotics and Automation*, Apr. 2007.

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

[31] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proceedings of the ninth European Conference on Computer Vision*, May 2006.

[32] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of European Conference on Computer Vision*, May 2006.

[33] ——, "Fusing points and lines for high performance tracking," in *Proceedings of IEEE International Conference on Computer Vision*, Oct. 2005.

[34] P. Doherty, P. Haslum, F. Heintz, T. Merz, T. Persson, and B. Wingman, "A distributed architecture for intelligent unmanned aerial vehicle experimentation," in *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, 2004.

[35] M. Wzorek, G. Conte, P. Rudol, T. Merz, S. Duranti, and P. Doherty, "From motion planning to control - a navigation framework for an autonomous unmanned aerial vehicle," in *21th Bristol UAV Systems Conference*, Apr. 2006.

[36] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the fastslam algorithm," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, May 2006.

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

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

Rickard Karlsson was born 1970 in Örebro, Sweden. He received the M.Sc. degree in Applied Physics and Electrical Engineering in 1996, and a Ph.D. in Automatic Control in 2005, both from Linköping university, Linköping, Sweden. Between 2005–2007 employed as research associate at the Automatic Control group in Linköping. From 2007 he has been at NIRA Dynamics AB. His research interests include positioning and tracking applications mainly using particle filters.

Thomas B. Schön was born in Sweden in 1977. He received the B.Sc. degree in Business Administration and Economics in Feb. 2001, the M.Sc. degree in Applied Physics and Electrical Engineering in Feb. 2001 and the Ph.D. degree in Automatic Control in Feb. 2006, all from Linköping University, Linköping, Sweden. He has held visiting positions at the University of Cambridge (UK) and the University of Newcastle (Australia). His research interests are mainly within the areas of signal processing, sensor fusion and system identification, with applications to the automotive and aerospace industry. He is currently a Research Associate at Linköping University.

David Törnqvist is a PhD student at Automatic Control, Linköping University, since 2003. In 2006 he received his Licentiate degree in automatic control and in 2003 he received his Master of Science in Communication and Transport Engineering, both from Linköping University. In his research, the main focus is on statistical estimation and detection methods. Applications studied is Simultaneous Localization and Mapping (SLAM) as well as disturbances to navigation sensors. He has held visiting positions at University of Sydney (Australia) and University of California, Berkeley (USA).

Gianpaolo Conte is a PhD student at the Department of Computer and Information Science, Linköping University, Sweden. He obtained the Licentiate degree at the same University and the Aerospace Engineering degree at Turin Polytechnic. He is interested in navigation and control problem for UAVs. He is also working on the development of Micro Aerial Vehicles platforms.

Fredrik Gustafsson is professor in Sensor Informatics at Department of Electrical Engineering, Linköping University, since 2005. He received the M.Sc. degree in electrical engineering 1988 and the Ph.D. degree in Automatic Control, 1992, both from Linköping University. During 1992-1999 he held various positions in automatic control, and in 1999 he got a professorship in Communication Systems. In 2004, he was awarded the Arnberg prize by the Royal Swedish Academy of Science (KVA) and in 2007 he was elected member of the Royal Academy of Engineering Sciences (IVA). His research interests are in stochastic signal processing, adaptive filtering and change detection, with applications to communication, vehicular, airborne and audio systems, where the current focus is on sensor fusion algorithms for navigation and tracking problems. He was an associate editor for IEEE Transactions of Signal Processing 2000-2006 and is currently associate editor for EURASIP Journal on Applied Signal Processing and International Journal of Navigation and Observation.