EM-SLAM with Inertial/Visual Applications

Zoran Sjanic, Martin A. Skoglund, Fredrik Gustafsson *Fellow, IEEE*

***Abstract*—The general Simultaneous Localisation and Mapping (SLAM) problem aims at estimating the state of a moving platform simultaneously with map building of the local environment. Current state-of-the-art methods such as [1] relies on Nonlinear Least-Squares (NLS) batch formulations with structure exploitation for memory efficiency and speed. We investigate the Expectation Maximisation (EM) algorithm for solving a generalized version of the NLS problem. This EM-SLAM algorithm solves two simpler problems iteratively yielding a low computational complexity. The iterations switch between state estimation, which can use any state space smoother, and map estimation, where a quasi-Newton method is suggested. The proposed method is evaluated in real experiments and also in simulations on a platform with a monocular camera attached to an inertial measurement unit. The results show that EM-SLAM has much lower computational complexity than NLS while maintaining comparable accuracy.**

***Index Terms*—SLAM, Expectation-Maximisation, Sensor Fusion, Computer Vision, Inertial Sensors, Vision-Aided Navigation.**

I. Introduction

The aim in Simultaneous Localisation and Mapping (SLAM) is to estimate a moving platform's position and orientation while mapping the observed environment. A strong trend in SLAM algorithm research is (incremental) batch optimisation which usually solves some form of Nonlinear Least-Squares (NLS) problem [1]–[5]. These often suffers from poor complexity scaling, usually quadratic in batch length. One popular approach is GraphSLAM [6]–[8], where the idea is that the map is implicitly defined through a set of robot poses and the sensor measurements without the need for any explicit map representation. Hence, the only parameters are the poses at the cost of quadratic scaling in batch length, rather than preferably map size [9]. To remedy this many heuristics have been proposed, see e.g., [10], [11]. The graphs are constructed from either incremental pose constraints due to odometry or constraints between arbitrary poses due to loop closing based on sensor data. Pose graphs are natural when the sensor model is locally invertible as in the case of laser scanners [12] or stereo vision [10].

Methods based on camera-only, such as structure from motion (SfM) [13], [14] or Bundle Adjustment (BA) [15], [16], have been known in the computer vision community for quite some time. The estimated structure and motion have unknown universal scale, since the monocular camera suffers from depth ambiguity. In other words, given a motion of the camera, we cannot say if the velocity was large and the scene was far away or if the velocity was low and the scene was close to the camera. One way to resolve the universal scale is to add some kind of velocity measurement, and the inertial measurement unit (IMU), measuring accelerations and angular velocities, is one such way [17]–[21].

There are only a few methods with monocular vision and inertial sensors that use pose graphs. For instance, [4], [9] use view-based matching on whole images combined with odometry and the obtained relative poses are used to construct the graph. A landmark-free sliding window EKF for visual odometry with IMU support is proposed in [22]. In the same spirit [23] uses epipolar geometry and trifocal constraints which implicitly encodes landmarks, while the IMU is integrated between image frames for complexity reduction. However, these methods are structure-less and therefor require other additional computation, if the map is explicitly sought.

Here, we propose a Maximum Likelihood (ML) formulation which is based on the Expectation Maximisation (EM) algorithm [24]. Contrary to GraphSLAM, both the pose and the map are explicitly estimated as in ordinary NLS. The EM framework allows for moving the dominating complexity of the ML solution to scale linearly with the batch length and the map size, rather than quadratically, while still retaining an accuracy comparable to NLS. This is illustrated in a qualitative analysis as well as in experiments. The method is evaluated on, but not restricted to, a system with a monocular camera and IMU sensors. In an EM setting, so called latent, or hidden, variables are introduced in order to solve difficult ML problems. This is achieved by splitting the problem into two simpler problems, one where expectation with respect to the conditional density of the latent variables has to be calculated and one where a certain function needs to be maximised with respect to the parameters. These two steps are then repeated until convergence. In EM-SLAM, the map is viewed as the unknown parameter and the platform states, such as position and orientation, are considered to be the latent variables. As a simplified and intuitive motivation for this separation we can consider two simpler problems; one with known map and the other one with known trajectory and orientation. The first problem is then simply the navigation problem with known landmarks. The second problem is known as the triangulation problem, i.e., finding the landmark positions given the known platform positions and camera observations. See e.g., [25] for an example of triangulation application. Each of these problems are rather straightforward to solve separately, but hard to solve combined. By separating the variables in the proposed way, we split the SLAM problem into the above-mentioned two simpler problems. In the conditional expectation step, i.e., navigation, the latent variables are assumed to have Gaussian distribution and that they can be well approximated with an Extended Rauch-Tung-Striebel (E-RTS) smoother, [26].

Zoran Sjanic, Martin A. Skoglund and Fredrik Gustafsson are with the Division of Automatic Control, Department of Electrical Engineering, Linköping University SE-581 83 Linköping, Sweden Email: {zoran, ms, fredrik}@isy.liu.se

The authors gratefully acknowledge funding from the Vinnova Industry Excellence Center LINK-SIC.

However, for other distribution assumptions on the latent variables any appropriate state space smoother can be used. The maximisation step, i.e., triangulation, is solved using a quasi-Newton method. The proposed method is compared with the NLS method, which can be seen as a straightforward ML formulation where both the state sequence and the map are seen as parameters. This reference method is solved using the Levenberg-Marquardt algorithm [27]. The comparison is done for both the performance of the estimation and for the complexity of each approach on both simulations and real data experiment.

The paper is organised as follows; in Section II the Expectation Maximisation algorithm is explained in more detail, and its application to SLAM is described in Section III. The dynamical and measurement models specific to visual/inertial SLAM are introduced in Section IV and NLS-SLAM is explained in Section V. Complexity comparisons between EM-SLAM, NLS-SLAM and GraphSLAM are discussed in Section VI, and a brief explanation about obtaining an initial estimate for the landmarks is given in Section VII. Finally, results, conclusions and future work are discussed in Section VIII and Section IX.

II. Expectation Maximisation

Maximum likelihood in its basic form is a batch method which takes a set of K observations, $Y = \{y_1, \ldots, y_K\}$, and aims at maximising the measurement likelihood, $p_\theta(Y)$, w.r.t., the unknown parameter θ as

$$\hat{\theta}^{ML} = \arg\max_\theta \, p_\theta(Y) = \arg\min_\theta \, -\log p_\theta(Y). \tag{1}$$

The maximisation of (1) can be very difficult and the key idea with Expectation Maximisation is to consider the joint likelihood, $p_\theta(Y, X)$, where $X = \{x_1, \ldots, x_K\}$ are latent variables. Then, by splitting this density into two coupled, and hopefully easier, problems the parameters and the latent variables can be solved for in an iterative manner. The first step is the *Expectation step*, commonly denoted *E-step*, where the expectation of the joint log-likelihood, $\log p_\theta(Y, X)$, with respect to the density of the latent variable conditioned on all the measurements, $p_{\theta_k}(X|Y)$, is computed. The expectation $\mathbf{E}_{\theta_k}\{\log p_\theta(X, Y)|Y\}$ will be a function, called $\mathcal{Q}(\theta, \theta_k)$, of the parameter vector θ as

$$\mathcal{Q}(\theta, \theta_k) = \int p_{\theta_k}(X|Y) \log p_\theta(Y, X) \, \mathrm{d}X. \tag{2}$$

Note that the conditional density of the latent variables, $p_{\theta_k}(X|Y)$, is computed using the estimate of the parameters from the previous iteration, called k, θ_k, which is also emphasised in the notation. In the *Maximisation step* or *M-step*, the $\mathcal{Q}$ function, obtained in the E-step, is maximized with respect to the parameter θ, obtaining new estimate θ_{k+1}. These two steps are repeated until some convergence criterion is met, usually when the change in the parameter or the likelihood value is below a certain threshold.

For an explanation of the EM algorithm applied to dynamical systems see e.g., [28], and how it can be used in system identification is exemplified in [29], where a particle smoother is used to calculate the conditional expectation in the E-step. In [30], [31] nonlinear dynamical models are treated using EM, and the E-step is calculated using an Extended Kalman smoother, which is the same approach that will be used here. All these EM variants are formulated as batch methods, but there are also online EM methods which typically use sequential Monte Carlo and stochastic approximation methods [32], [33]. However, in these online approaches it is assumed that either the joint log-likelihood belongs to the exponential family, or that the state is low dimensional and can be well approximated with particle methods. In our case these assumptions are not met which will require other approximations to be applied.

III. EM-SLAM

We formulate the visual/inertial SLAM problem by defining a state space model as

$$x_t = f(x_{t-1}, u_t, w_t), \tag{3a}$$

$$y_t = h_t(x_t, \theta) + e_t, \tag{3b}$$

where the measurement noise, e_t, is considered white and Gaussian with mean zero and covariance R, while the process noise, w_t, is considered white with mean zero and covariance Q. The state transition function is denoted f, and $\{u_t | t \in \{1, \ldots, K\}\}$, where K is the total number of signal time instances, are considered to be inputs given by the inertial sensors from which three-dimensional position, velocity and orientation, $x_t = [p_t^T, v_t^T, q_t^T]^T$, $x_t \in \mathbb{R}^{10}$, are computed. The measurements y_t are the two-dimensional camera measurements, i.e., features extracted from images. In general, an IMU has a higher sampling rate than a camera and a multi-rate system model is obtained. This can be denoted as $\{y_t | t \in G \subseteq \{1, \ldots, K\}\}$, where G contains indices to camera observation instances, and the cardinality is $|G| = N$, i.e., there are N images. Here, we also assume that the IMU and the camera are synchronised in time. The measurement function, h_t, relates measurements, states and parameters. Usually, only a part of all the landmarks are measured in each image, and that number is denoted N_t, i.e., $y_t \in \mathbb{R}^{2\,\mathsf{N}_t}$. The parameter vector θ consists of landmark coordinates in three dimensions, i.e., $\theta \in \mathbb{R}^{3M}$ if there are M landmarks. The models in (3) will be defined in detail in Section IV.

The most significant difference between the proposed EM-SLAM formulation and batch SLAM is that the map θ is treated as a time-independent parameter, while the vehicle state x_t is a constant size vector. This separation is the key to allow for an efficient algorithm since the computation of the conditional expectation, E-step, scales linearly in the batch length while the map estimation, M-step, scales linearly in the number of landmark measurements (which will roughly increase linearly with the batch length and the map size). Analysis of the complexity will be treated in more detail in Section VI.

The conditional expectation step is assumed to be well approximated by an Extended Rauch-Tung-Striebel (E-RTS) smoother. E-RTS is a straightforward modification of the standard RTS smoother, [26], by using the Extended Kalman

Filter instead of the Kalman Filter in the forward filtering step, while the backward smoothing step is the same as in the original RTS smoother.

The state space formulation above constitutes the basis for the ML formulation that is naturally put into EM setting, i.e., it is straightforward to define the joint likelihood $p_\theta(X, Y)$. Here the platform states, X, are considered to be latent variables. By using the Markov properties this density can be written as

$$p_\theta(Y, X) = \prod_{t=1}^{K} p_\theta(y_t|x_t)p(x_t|x_{t-1}). \tag{4}$$

Notice that the process model does not depend explicitly on the parameter θ, which will simplify the calculations significantly, as will be shown in the next section.

Next, both the E-step and the M-step will be explained in detail, with all derivations and approximations used.

A. E-step

In each E-step, the landmark locations θ_k, obtained in the previous M-step (i.e., iteration k), are fixed and used to estimate the platform's state $x_t,\ t = 1, \ldots, K$. Given the joint likelihood from (4), the expectation step produces a function template in the following form

$$\mathcal{Q}(\theta, \theta_k) = \mathbf{E}_{\theta_k}\left\{\log\left[\prod_{t=1}^{K} p_\theta(y_t|x_t)p(x_t|x_{t-1})\right]\Big|Y\right\}, \tag{5}$$

where the measurement likelihood is given by

$$p_\theta(y_t|x_t) = p_\theta(e_t) = p_\theta(y_t - h_t(x_t, \theta)), \tag{6}$$

and the state transition density, $p(x_t|x_{t-1})$, does not depend on θ. Assuming that the likelihood has a Gaussian distribution, the expectation (5) becomes

$$\begin{aligned}\mathcal{Q}(\theta, \theta_k) &= \text{const.} - \\ &\mathbf{E}_{\theta_k}\left\{\sum_{t=1}^{K}\frac{1}{2}\|y_t - h_t(x_t, \theta)\|^2_{R^{-1}} + \log p(x_t|x_{t-1})\Big|Y\right\} \\ &= -\sum_{t\in G}\mathbf{E}_{\theta_k}\left\{\frac{1}{2}\|y_t - h_t(x_t, \theta)\|^2_{R^{-1}}\Big|Y\right\} + \text{const.}\end{aligned} \tag{7}$$

where all the terms not depending on θ are lumped into a constant term, which will not affect the optimisation in the subsequent step. Notice also that the summation limit is changed to only include the time instances for the images, resulting in N terms in the sum. Due to the nonlinear nature of the measurement function, see Section IV-B, there is no closed form solution. Thus, some approximations are necessary, and one such is

$$\begin{aligned}\mathcal{Q}(\theta, \theta_k) \approx \text{const.} - \frac{1}{2}\sum_{t\in G}\Big(&\|y_t - h_t(\hat{x}_{t|K}, \theta)\|^2_{R^{-1}} + \\ &\text{Tr}(R^{-1}\nabla_x h_t(\hat{x}_{t|K}, \theta)P^s_{t|K}(\nabla_x h_t(\hat{x}_{t|K}, \theta))^T)\Big).\end{aligned} \tag{8}$$

Here, $\hat{x}_{t|K}$ is the smoothed estimate of the latent variable and $P^s_{t|K}$ is its covariance. The smoothed estimate is obtained with an E-RTS smoother in the E-step summarised in Algorithm 1.

Algorithm 1 E-step

Input: measurements $\{y_t|t \in G \subseteq \{1, \ldots, K\}\}$, inputs $\{u_1, \ldots, u_K\}$, covariance matrices Q and R, initial state and its covariance, $x_0,\ P_0$, parameter estimate θ_k
Output: $\mathcal{Q}(\theta, \theta_k)$

$\hat{x}_{0|0} := x_0$
$P_{0|0} := P_0$
for $t = 1 : K$ **do**
  $\hat{x}_{t|t-1} := f(\hat{x}_{t-1|t-1}, u_t)$
  $P_{t|t-1} := F_{t-1}P_{t|t-1}F^T_{t-1} + Q$
  if Image available **then**
    $\hat{x}^0_{t|t-1} := \hat{x}_{t|t-1}$
    $P^0_{t|t-1} := P_{t|t-1}$
    for $j = 1 : \mathsf{N}_t$ **do**
      $S^j := H^j_t P^{j-1}_{t|t-1}(H^j_t)^T + R^{jj}$
      $K^j := P^{j-1}_{t|t-1}(H^j_t)^T(S^j)^{-1}$
      $P^j_{t|t} := P^{j-1}_{t|t-1} - K^j H^j_t P^{j-1}_{t|t-1}$
      $\hat{x}^j_{t|t} := \hat{x}^{j-1}_{t|t-1} + K^j(y^j_t - h_t(\hat{x}^{j-1}_{t|t-1}, \theta^j_k))$
    end for
    $\hat{x}_{t|t} := \hat{x}^{\mathsf{N}_t}_{t|t}$
    $P_{t|t} := P^{\mathsf{N}_t}_{t|t}$
  end if
  Where $F_{t-1} = \frac{\partial}{\partial x}f(x, u_t, w_t)|_{x=\hat{x}_{t-1|t-1}, w_t=0}$, and $H^j_t = \frac{\partial}{\partial x}h_t(x, \theta^j_k)|_{x=\hat{x}^{j-1}_{t|t-1}}$ are the Jacobians of the expressions in (3) while θ^j_k picks out the component corresponding to measurement y^j_t.
end for
$P^s_{K|K} := P_{K|K}$
for $t = K : 2$ **do**
  $S_{t-1} := P_{t-1|t-1}F^T_{t-1}P^{-1}_{t|t-1}$
  $\hat{x}_{t-1|K} := \hat{x}_{t-1|t-1} + S_{t-1}(\hat{x}_{t|K} - \hat{x}_{t|t-1})$
  $P^s_{t-1|K} := P_{t-1|t-1} + S_{t-1}(P^s_{t|K} - P_{t|t-1})S^T_{t-1}$
end for
Assemble $\mathcal{Q}(\theta, \theta_k)$ according to (8).

The notation $\hat{x}_{t|s}$ denotes the estimate of the state x_t at time t given all the measurements up to time s, and likewise for the covariance P. Note also that the measurement update in the forward pass of E-RTS can be processed sequentially since the measurements are assumed to be independent. The trace term can be thought of as a regularisation term to compensate for the usage of the estimated latent variables instead of the true ones. If the true ones have been used that term would vanish and only the nonlinear least squares part had to be solved. Note also that the terms in the sum are nonzero only for the observed landmarks per time step. See Appendix for derivation of (8).

B. M-step

In each M-step the platform state is fixed and the location of the landmarks are triangulated. Maximisation (minimisation) of the $\mathcal{Q}$ ($-\mathcal{Q}$)-function can be done using standard optimisation software. As for our particular setting, the function to be minimised is a nonlinear function of the parameters and

Algorithm 2 M-step (Quasi-Newton minimisation method with BFGS Hessian update)

Input: Function template $\mathcal{Q}(\theta, \theta_k)$, initial parameters θ_k, initial inverse Hessian approximation $B_0 = \lambda I$, $\lambda > 0$, termination threshold ε.
Output: θ_{k+1}.

$i := 0$
terminate := **false**
$\theta^i := \theta_k$
while not *terminate* **do**
  Compute search direction:
  $p_i := -B_i \nabla_\theta \mathcal{Q}(\theta^i, \theta_k)$
  Update the parameter:
  $\theta^{i+1} := \theta^i + \alpha_i p_i$
  where α_i is the step length computed by line search ensuring decrease in cost
  Compute:
  $s_i := \theta_{i+1} - \theta_i$
  $r_i := \nabla_\theta \mathcal{Q}(\theta^{i+1}, \theta_k) - \nabla_\theta \mathcal{Q}(\theta^i, \theta_k)$
  Update the inverse Hessian
  $B_{i+1} := \left(I - \frac{s_i r_i^T}{r_i^T s_i}\right) B_i \left(I - \frac{r_i s_i^T}{r_i^T s_i}\right) + \frac{s_i s_i^T}{r_i^T s_i}$
  if Some termination criterion depending on ε is met **then**
    terminate := **true**
  else
    $i := i + 1$
  end if
end while
$\theta_{k+1} := \theta^{i+1}$

nonlinear methods need to be used. We use a quasi-Newton method called BFGS, [27], since it is quite efficient, but other choices are also possible. In this method, the inverse Hessian of the function to be optimised is recursively approximated using gradient information. Suitable termination criteria may be that the magnitude of change in parameter values, the gradient, or the magnitude of the cost function decrease is sufficiently small. The BFGS algorithm is summarised in Algorithm 2.

IV. Models

In this section the models in (3) will be specified. The sensors of interest are monocular camera and 6-DOF inertial sensors, i.e., gyroscopes and accelerometers, contained in a single sensor package. To reduce the state and parameter space the inertial sensors are considered as inputs to a process model. A minimal 3D point landmark parametrisation is used and its measurement function is given by the pinhole projection model.

A. IMU Parametrisation

The models for the gyroscopes and accelerometers are simple as they are only considered to be inputs to the process model. The gyroscope signals are denoted $u^\omega = [u_x^\omega, u_y^\omega, u_z^\omega]^T$ where the subscript refers to each axis of the body frame. Similarly the accelerometer signals are denoted $u^\mathsf{a} = [u_x^\mathsf{a}, u_y^\mathsf{a}, u_z^\mathsf{a}]^T$ which are also given in the sensor body frame. A discretised process model for the position, velocity and rotation, $[p_t^T, v_t^T, q_t^T]^T$, in the local, inertial, navigation frame is then,

$$p_t = p_{t-1} + T v_{t-1} + \frac{T^2}{2} \mathsf{R}^T(q_{t-1}) \left(u_t^\mathsf{a} + g^b + w_t^\mathsf{a}\right), \quad (9a)$$

$$v_t = v_{t-1} + T \mathsf{R}^T(q_{t-1}) \left(u_t^\mathsf{a} + g^b + w_t^\mathsf{a}\right), \quad (9b)$$

$$q_t = \exp\left(\frac{T}{2} S_\omega(u_t^\omega + w_t^\omega)\right) q_{t-1}, \quad (9c)$$

where the T denotes the sampling interval, $\mathsf{R}(q_t)$ is a rotation matrix parametrisation of the unit quaternion $q_t = [q_t^0, q_t^1, q_t^2, q_t^3]^T$ which describes the rotation from navigation to body frame, $g^b = \mathsf{R}(q_t) g^n$, is the gravity expressed in the body frame, $g^n = [0, 0, -g]$ is the local gravity vector expressed in the inertial frame where $g \approx 9.82$ and $\exp(\cdot)$ is here considered as the matrix exponential. The noise terms are assumed Gaussian and independent $[(w_t^\mathsf{a})^T, (w_t^\omega)^T]^T = w_t \sim \mathcal{N}(0, diag(Q_\mathsf{a}, Q_\omega))$. The skew-symmetric matrix

$$S_\omega(u) = \begin{bmatrix} 0 & -u_x & -u_y & -u_z \\ u_x & 0 & u_z & -u_y \\ u_y & -u_z & 0 & u_x \\ u_z & u_y & -u_x & 0 \end{bmatrix}, \quad (10)$$

parametrises the quaternion dynamics. This parametrisation is very similar to reduced-dimension observers in [34].

B. Camera Measurements

The monocular camera is modeled as a standard pinhole camera, see cf. [35]. The camera calibration matrix and lens distortion were estimated prior to usage. Since the calibration and distortion are known the undistorted pixels can be premultiplied with the inverse of the camera matrix, thus the camera then works as a projective map in Euclidean space, $P : \mathbb{R}^3 \to \mathbb{R}^2$. The projection is defined as $P([X, Y, Z]) = [X/Z, Y/Z]$ and the Z coordinate is assumed positive and nonzero since otherwise the point would be behind the camera. Then a normalised camera measurement, $y_t = [u_t, v_t]^T$, of a landmark, m, at time t is

$$y_t = P(\mathsf{R}(q_t)(m - p_t)) + e_t, \quad (11)$$

which relates the pose (position and orientation) of the camera to the 3D location of the point. The measurement noise is assumed i.i.d. Gaussian, $e_t = [e_t^u, e_t^v]^T \sim \mathcal{N}(0, R_f)$. The correspondence variables at time t, c_t^i, encode the measurement-landmark assignment, $y_t^i \leftrightarrow m^j$, which gives a subset of all M landmarks at time t, $\mathsf{M}_t = \{m^j\}$, $j \in \{1, \ldots, M \,|\, c_t^i = j\}$. At time t the stacked measurement equation is then

$$\underbrace{\begin{bmatrix} u_t^1 \\ v_t^1 \\ \vdots \\ u_t^{\mathsf{N}_t} \\ v_t^{\mathsf{N}_t} \end{bmatrix}}_{y_t^{\text{cam}}} = \underbrace{\begin{bmatrix} P(\mathsf{R}(q_t)(m^{c_t^1} - p_t)) \\ \vdots \\ P(\mathsf{R}(q_t)(m^{c_t^{\mathsf{N}_t}} - p_t)) \end{bmatrix}}_{h_t(x_t, \mathsf{M}_t)} + \underbrace{\begin{bmatrix} e_t^{1u} \\ e_t^{1v} \\ \vdots \\ e_t^{\mathsf{N}_t u} \\ e_t^{\mathsf{N}_t v} \end{bmatrix}}_{e_t^{\text{cam}}}, \quad (12)$$

where c_t^i denotes the index of the corresponding landmark and N_t is the number of camera measurements at time t and $e_t^{\text{cam}} \sim \mathcal{N}(0, R_{\text{cam}})$. R_{cam} is a diagonal matrix since all the measurements are assumed to be mutually independent. In this work the correspondences are assumed to be correctly solved in the initialization step (see Section VII) but in practice there will always be outliers of some kind. This is a strong assumption which should be treated carefully since faulty associations will bias the SLAM estimate. Interesting approaches to data association were exploited in e.g., [36], [37] which both make use of the EM algorithm to estimate correspondences.

V. Nonlinear Least-Squares

Another way of solving the ML SLAM problem is to consider all the interesting parameters explicitly instead of having position, velocity and orientation as hidden variables. In this case the parameter vector θ will consist of all unknown parameters, that is landmarks, accelerations in navigation frame and rate gyros. The dynamics for the velocity and position is in this case used as explicit constraints. In this setting it is also possible to include biases for accelerations and angular rates as parameters, which was avoided in the EM formulation. This is because the problem greatly simplifies if the parameters affect only the measurement relation, as already explained in Section III. Note however, that these extra terms can be put in the state vector. The measurement models for accelerations and angular rates are then

$$y_t^a = \mathsf{R}(q_t)(a_t - g^e) + b_a + e_t^a, \tag{13a}$$
$$y_t^\omega = \omega_t + b_\omega + e_t^\omega, \tag{13b}$$

and camera measurements are defined as in (12)

$$y_t^{\text{cam}} = h_t(p_t, q_t, \mathsf{M}_t) + e_t^{\text{cam}}. \tag{14}$$

The unknown parameters are then accelerations, $a_{1:N}$, angular rates, $\omega_{1:N}$, initial velocity v_0, acceleration bias, b_a, angular velocity bias, b_ω, and landmark positions m. Notice that in this setting the number of IMU measurements is the same as the number of images. For the multi-rate models considered here the IMU signals between camera measurements can simply be averaged as

$$\bar{y}_t^{a,\omega} = \frac{1}{G_t - G_{t-1}} \sum_{s=G_{t-1}}^{G_t} y_s^{a,\omega}. \tag{15}$$

where G_i denotes the i:th element of G (defined in Section III) and $\bar{y}_t$ is the averaged measurement used at the image time instance t. Under the assumption that all noises are Gaussian and white, i.e., $e_t^i \sim \mathcal{N}(0, R_i)$, the corresponding negative log-likelihood becomes

$$\begin{aligned} -\log p_\theta(Y) = \sum_{t=1}^{N} & \|\bar{y}_t^a - \mathsf{R}(q_t)(a_t - g^e) - b_a\|^2_{R_a^{-1}} + \\ & \|\bar{y}_t^\omega - \omega_t - b_\omega\|^2_{R_\omega^{-1}} + \\ & \|y_t^{\text{cam}} - h_t(p_t, q_t, \mathsf{M}_t)\|^2_{R_{\text{cam}}^{-1}}. \end{aligned} \tag{16}$$

where $\theta = [a_{1:N}^T, \omega_{1:N}^T, v_0^T, b_a^T, b_\omega^T, m^T]^T$, q_t is a function of $\omega_{1:t}$ and p_t is a function of v_0 and $a_{1:t}$. The ML problem can now be formulated as

$$\hat{\theta}^{ML} = \arg\min_\theta \; -\log p_\theta(Y) \tag{17a}$$

$$\text{subject to} \quad \begin{bmatrix} p_t \\ v_t \end{bmatrix} = F^t \begin{bmatrix} p_0 \\ v_0 \end{bmatrix} + \sum_{i=1}^{t} F^{i-1} B a_i, \tag{17b}$$

$$F = \begin{bmatrix} I_3 & TI_3 \\ 0 & I_3 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{T^2}{2} I_3 \\ TI_3 \end{bmatrix},$$

$$q_t = \left[\prod_{k=1}^{t} \exp\left(\frac{T}{2} S_\omega(\omega_k) \right) \right] q_0. \tag{17c}$$

The constraints can actually be removed by expanding them and substituting them into the cost function giving an unconstrained problem. This problem is solved with e.g., standard Levenberg-Marquardt solver. The estimate obtained in this way will be used to compare to the estimate obtained with the EM-SLAM method.

Note that this particular choice of variables is only one of many possibilities to formulate the NLS problem. Another alternative is to parameterise motion as only poses [5], [38] and parameterisation has in general a great impact on both computational complexity and estimation result. For example, different structural properties of the problem can be utilised, like sparsity of the Jacobian. In many of those cases the complexity is comparable to EM-SLAM (and very similar to GraphSLAM), but the problem can be somewhat non-physically formulated. As an example, if quaternions are used as optimisation variables (which is preferred to Euler angles in three dimensions due to gimbal lock problem) the error term between two quaternions is not a new quaternion [39], since they reside in a $SO(3)$ and not in a Euclidian space (due to the constraint $q^T q = 1$). This implies that some kind of compensation, e.g., normalisation (in each optimisation iteration) or constrained formulation, must be done in order to solve that kind of problem, which of course increases complexity. Our motivation to compare with the setup in (17) is twofold; first, the error terms should have a physical interpretation, i.e., sensor noise; second, the optimisation variables should reside in Euclidean space due to the nature the iterative search methods.

VI. Computation Complexity

The main difference between NLS and EM approach is the number of parameters in the optimisation part. While NLS has both landmarks and platform's motion as parameters, EM considers the motion as latent variables. Seen the other way around, the ML problem in (1) can be considered as a marginalised version of (16), where motion is integrated out. GraphSLAM, on the other hand, marginalises landmarks and solves the problem only in the motion variables. There are very few examples of GraphSLAM with inertial and monocular camera sensors, due to the non-invertability of the camera measurement equation, but for the comparison's sake we will give some complexity analysis for GraphSLAM also. For densely observed environments, where the number

of landmarks grows much slower than number of motion parameters, EM-SLAM has potential to have lower complexity than GraphSLAM as will be shown below. A small numerical simulation example will be given for execution time comparison. It should be pointed out that, in our analysis, we are not interested in the exact number of operations, since it is implementation dependent in many cases. For example, matrix multiplications might take into account the structure of the matrices and thus decrease the actual number of operations needed. Also, there are many efficient and mature implementations of various batch SLAM methods available, see e.g., [1], [40], which handles impressively huge data sets. Instead, we will compare different methods' inherent complexity, i.e., the number of terms that need to be evaluated in one iteration during iterative optimisation, with the order of magnitude, denoted with $\mathcal{O}$-notation. That is, $\mathcal{O}(n) = \mathcal{O}(an)$, where a is a constant independent of n. It is also assumed that all functions are analytical and that their gradients can be evaluated with constant cost independent of the number of landmarks, measurements or image frames.

In our analysis we will use following notation; the number of landmarks is M, the number of time steps when landmarks are observed is N, i.e., it is the number of image frames. The total number of landmark observations (or measurements) is N_c, which is defined as

$$N_c = \sum_{t \in G} \mathsf{N}_t, \tag{18}$$

and is linear in the number of camera frames. Note that in the case when all the landmarks are observed the whole time, we have $N_c = NM$. Define further the average number of measurements per camera time step as $\bar{N}_c^N = N_c/N$ and the average number of measurements per landmark as $\bar{N}_c^M = N_c/M$.

A. EM-SLAM

Each iteration of EM-SLAM requires one E-step and one M-step. For the E-step, we assume that the most demanding operation in E-RTS is the measurement update in the forward EKF pass. An IMU usually has a higher sampling rate than the camera, but these data are only used in the time update steps between camera measurements, and recall that the size of the state is 10, which is constant. Also, in the same way, the backward pass of the E-RTS consists of the constant size matrices and vectors independent of the number of camera measurements. The number of operations in the measurement update step is assumed to be proportional to the number of camera measurements in that step. This assumption can be made due to the mutual independence of the measurements, so that the update can be performed sequentially, see e.g., [41]. Since there are N measurement update steps and the average number of measurements per time step is $\bar{N}_c^N$, we have $\mathcal{O}(\bar{N}_c^N N)$ in total, i.e., $\mathcal{O}(N_c)$, using the definition of average number of measurements per time step. Note that if the time update and the backward pass are also considered to contribute to the complexity, their contribution is a constant, C, times the number of time steps N. This constant depends on the number of states and the sampling rate of the IMU but independent of the number of camera frames, landmarks and measurements. In that case we will have the average complexity of $\mathcal{O}((\bar{N}_c^N + C)N)$ which is still linear in N.

In the M-step, the main complexity lies in the calculation of the $\mathcal{Q}$-functions gradient with respect to parameters, θ, which is a sum of all individual gradients for each landmark in each frame where the landmark is observed. In average this will be the average number of measurements per landmark $\bar{N}_c^M$. Since there are M landmarks, the total complexity is $\mathcal{O}(\bar{N}_c^M M) = \mathcal{O}(N_c)$, exactly as the E-step. It is also worth mentioning that calculation of the trace term can be simplified by calculation only the diagonal elements of the matrix product used in the trace. This gradient is calculated a number of times during the minimisation procedure, so the total complexity is $\mathcal{O}(kN_c)$. This k is hard to estimate, and is highly dependent on the particular choice of optimisation routine and the implementation. For our particular setup using BFGS, we have empirically obtained an average of 10 M-step iterations in simulations and real data experiments, which is negligible in comparison to other sizes. This gives that the complexity per one iteration of EM-SLAM is $\mathcal{O}(N_c)$ in total.

B. NLS-SLAM

For NLS-SLAM the Jacobian of the loss function (17) is needed and it consist of derivatives of the acceleration measurement errors w.r.t., acceleration and angular rate parameters, derivatives of the angular rate measurement errors w.r.t., angular rate parameters and derivatives of the camera measurement errors w.r.t., acceleration, angular rate and landmark parameters.

The number of elements in the part of the Jacobian that contains acceleration errors derivatives comes from the number of acceleration and angular rates parameters, which both are N, and the number of acceleration measurements, which also is N. Since error terms in a certain time depend only on acceleration parameters from the same time there are N of these terms. The acceleration error is dependent on the rotation in the same time instance and the rotation can be obtained by integration of the angular rates up to that time, see (17c). This means that Jacobian's row for a certain measurement will have non-zero elements for all angular rates up to the time for that measurement. This implies that the number of elements is approximately $.5N^2$. Together it gives $\mathcal{O}(N^2+N)$ complexity for the first part of the Jacobian.

The angular rates error terms at a certain time instance depend only on the angular rates from the same time instance. Since there are N such terms in total the complexity is $\mathcal{O}(N)$ due to the same reason as for the acceleration errors. The last error terms are the camera measurement terms, and they depend on platform positions, rotations and landmark positions. Platform positions at a certain time can be integrated from accelerations up to that time, just as the rotations mentioned above, using (17b). This means that we need to use all the acceleration and angular rates parameters up to a time for the camera measurement in that time. Since there are N_c camera measurements in total, the approximate number of elements to

be evaluated is proportional to $.5N_cN$ for both accelerations and angular rates, giving the complexity $\mathcal{O}(N_cN)$. For the derivative w.r.t., landmarks the same reasoning as for the M-step in EM-SLAM can be applied, since these are the same errors depending on the same parameters, and the complexity of that part was $\mathcal{O}(\bar{N}_c^M M) = \mathcal{O}(N_c)$. Summing all terms from above, the total complexity for evaluating the Jacobian is $\mathcal{O}(N_cN + N^2 + N_c + N)$.

C. GraphSLAM

For GraphSLAM the Jacobian matrix is consisting of derivatives for positions and rotations with respect to each other. It is a symmetric matrix and has dimension N^2. The measurements are integrated (or marginalised) into the elements that represent the positions from which a measured landmark is observed from. In this way the landmark parameters are removed from the optimisation and the measurements are acting as constraints on relative positions of the platform with shared observed landmarks. For example, if the same landmark is observed from three positions, there will be non-zero entries in rows and columns that correspond to times for these positions. This implies that number of non-zero elements for each time step will be the number of all other elements that are connected via all shared landmarks. This number is the average number of measurements per landmark, N_c^M, in average. Since there are N rows (or columns), the total average complexity of GraphSLAM is $\mathcal{O}(\bar{N}_c^M N)$.

D. Comparisons

After this order of magnitude complexity analysis, it is interesting to compare the complexity of each method for different cases of batch length, N, and the number of landmarks, M. In Table I the dominating computation complexity for the three methods is compared for the cases where relative sizes of the batch length and the number of landmarks are varied. These are obtained by using the terms derived above and approximating them with dominating terms. From this table we can see that EM-SLAM has complexity that is always better than NLS-SLAM and better than, or the same as GraphSLAM for two cases, namely when the batch length is much larger than the number of landmarks and when they are approximately the same. Even with down-sampling, such as key-frames [11], the batch length grows with time which is a bottleneck [1]. The size of the map however grows with the explored space and this growth rate is typically controlled by the user.

In Figure 1 an empirical comparison between the calculation time for a NLS Jacobian and a $\mathcal{Q}$-function gradient together with E-RTS is illustrated as a function of the number of landmarks and the batch length. Here, the ratio between calculation times for one iteration is defined as $t_{\text{NLS}}/(t_{\nabla_\theta \mathcal{Q}} + t_{\text{E-RTS}})$. GraphSLAM is not simulated since we do not have any implementation for this particular setup. Here it is assumed that all the landmarks are measured in every image, i.e., the worst case. It can be seen that the relative calculation time grows approximately linearly with the number of images for a fixed number of landmarks. For an increasing number of landmarks, given a fixed number of images, the ratio seems to reach a constant value. These empirical results are in line with the theoretical calculations in Table I.

Complexity	Ratio	EM-SLAM	NLS-SLAM	GraphSLAM
Average	$N \gg M$	$\mathcal{O}(N_c)$	$\mathcal{O}(N_cN)$	$\mathcal{O}(N_cN)$
	$N \approx M$	$\mathcal{O}(N_c)$	$\mathcal{O}(N_cN)$	$\mathcal{O}(N_c)$
	$N \ll M$	$\mathcal{O}(N_c)$	$\mathcal{O}(N_cN)$	$\mathcal{O}(N_c/M)$
Worst case	$N \gg M$	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$
	$N \approx M$	$\mathcal{O}(NM)$	$\mathcal{O}(N^2M)$	$\mathcal{O}(N^2)$
	$N \ll M$	$\mathcal{O}(M)$	$\mathcal{O}(M)$	$\mathcal{O}(N^2)$

Table I: Comparison of the average and the worst case (all landmarks are measured all the time) complexity for evaluating elements needed in one iteration for the three methods with respect to the relative number of time steps and landmarks.

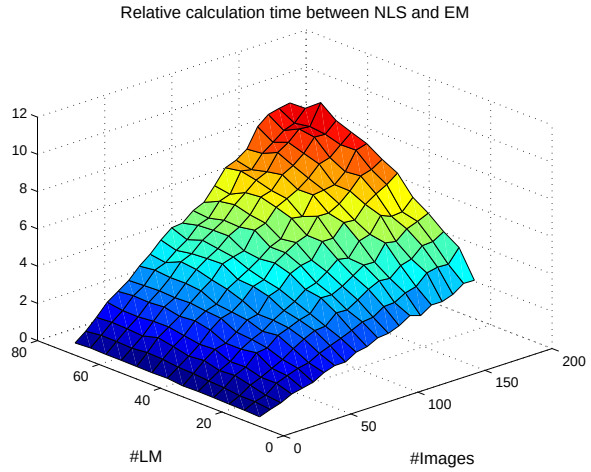


Figure 1: Relative calculation time between a NLS Jacobian and a gradient of the $\mathcal{Q}$-function together with E-RTS, $t_{\text{NLS}}/(t_{\nabla_\theta \mathcal{Q}} + t_{\text{E-RTS}})$.

Also, in Table II a result of a small simulated complexity benchmark example between EM-SLAM and NLS-SLAM is shown. In this example the number of landmarks is $M = 50$, number of time steps (images) is $N = 205$ and number of observations is $N_c = 4829$ so the number of landmarks is approximately 4 times smaller than number of time steps and all the landmarks are not observed the whole time. The total solution time is around 3 times lower for EM-SLAM method than NLS-SLAM method.

Furthermore, for large maps, the limited memory version L-BFGS could be used. It has footprint of $\mathcal{O}(kM)$ for both storage and computations, where k is the number of iterations.

Method	EM-SLAM	NLS-SLAM
Relative solution time [-]	1.0	3.1

Table II: Complexity comparison of the total solution time for each method through simulation. The number of landmarks is $M = 50$, the number of observations is $N_c = 4829$ and the number of time steps is $N = 205$.

VII. Obtaining an Initial Estimate

Both EM and NLS-SLAM need an initial value of the parameters in order to do the iterations. This initial value is also important for the performance of the methods, since both formulations are non-linear and non-convex. The initialisation can be performed by simply randomising parameter values but that can lead to solutions that are stuck in local minima. A better estimate of the initial values can be obtained by noting that the NLS-SLAM problem, defined in (17), is actually almost linear if rotations are fixed, [20]. In that case (13b) is not needed any more and (13a) is linear in parameters. For the landmark measurements consider the projection according to (11) which for fixed rotations can be rewritten as

$$\begin{bmatrix} [u_t \, \mathsf{R}_{3,:}(q_t) - \mathsf{R}_{1,:}(q_t)](m - p_t) \\ [v_t \, \mathsf{R}_{3,:}(q_t) - \mathsf{R}_{2,:}(q_t)](m - p_t) \end{bmatrix} = \begin{bmatrix} \mathsf{R}_{3,:}(q_t)(m - p_t)e_t^u \\ \mathsf{R}_{3,:}(q_t)(m - p_t)e_t^v \end{bmatrix}, \tag{19}$$

where $\mathsf{R}_{i,:}(q_t)$ denotes the i:th row of the rotation matrix. The only thing that makes this equation non-linear is the parameter dependent noise term. However this formulation leads to a well known Iterative Reweighted Least Squares (IRLS) method which is solved efficiently, see e.g., [42]. The accuracy of the estimate obtained in this way is dependent of the fixed rotations, but it still constitutes a much better initial value for the EM and NLS-SLAM then simply random values, see [43] for more details. The initial rotations can be obtained in several ways, for example simply by integrating rate gyros using (9c), or by some camera based method like 8-point, see e.g., [35]. The first method works quite fine if the gyro bias is small, while the latter one demands that the scene geometry is beneficial.

VIII. Results

Evaluation of the proposed method is carried out on both simulated and experimental data.

A. Simulations

Simulations give the ability to choose noise levels, correspondences, the true parameters and the true accuracy of the method. Monte Carlo (MC) simulations with 30 different measurement noise realisations have been performed in order to evaluate the performance of the proposed method and to compare EM-SLAM with NLS-SLAM. In Figure 2 the setup used for the simulations is illustrated and it is fixed for all the 30 simulations and the realisations of the measurement noise on accelerations, angular rates and camera observations are sampled from Gaussian distributions. The standard deviations for these distributions are $\sigma_a = 10^{-3}$ m/s^2, $\sigma_{\text{cam}} = 10^{-4}$ m and $\sigma_\omega = .5^\circ$/s.

Table III shows the average of the landmark estimation error and its standard deviation, averaged over all landmarks, for the two methods, while in Figure 3 the RMSE of the trajectory, for both methods, is plotted. In general, both methods have similar performance in terms of accuracy for both map and navigation states, although the NLS-SLAM has a smaller variance for the landmark estimate.

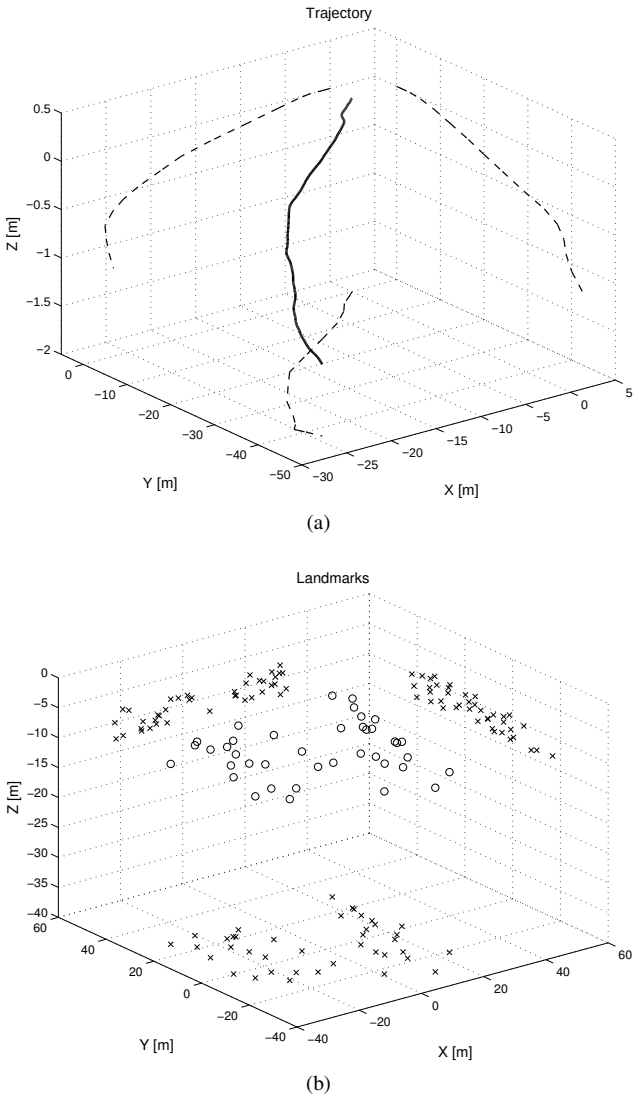


Figure 2: Environment setup used in Monte Carlo simulated experiments with simulated trajectory in (a) and simulated landmarks in (b). Both trajectory and landmarks are projected on all planes for a clearer view.

Method	**EM-SLAM**	**NLS-SLAM**
Mean $\|\hat{\theta} - \theta^*\|/\mathbf{dim}(\theta^*)$[m]	0.030	0.030
Std. Dev. [m]	$\pm$0.003	$\pm$0.0002

Table III: Mean estimation error and its standard deviation (1-σ), for the varying measurement noise (30 MC realisations). Note that θ contains only landmarks in this case.

B. Real Data Experiments

For the real data experiment, we use the data collected with the Yamaha Rmax UAV helicopter, owned by the Department of Computer and Information Science at Linköping University, during the flight trials performed in Revingehed, southern Sweden. The helicopter is equipped with GPS, IMU unit containing accelerometers and gyroscopes, monocular camera and facilities to record these data. In this experiment the IMU data rate was 20Hz and the image rate was 4Hz. For more

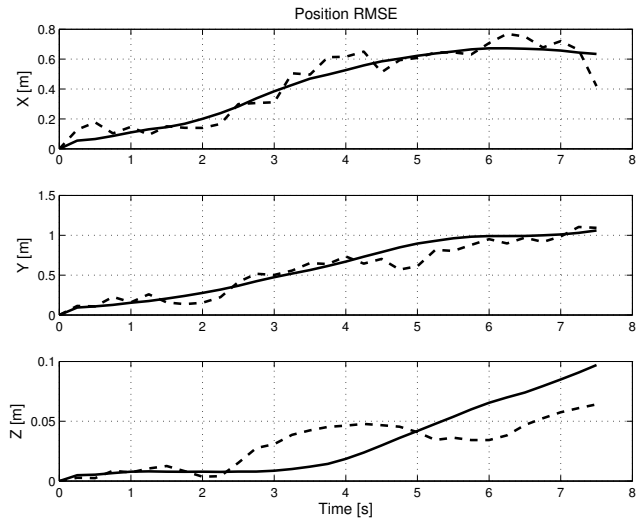


Figure 3: RMSE of EM-SLAM and NLS-SLAM estimated trajectories, EM in solid, NLS in dashed. 30 MC simulations are used.

Figure 4: Yamaha Rmax helicopter used for data collection in the real data experiment.

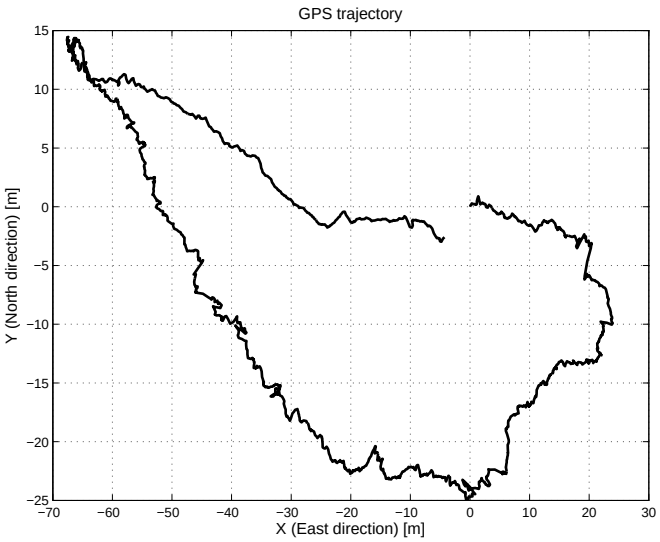


Figure 5: GPS trajectory of the helicopter in the East/North local plane.

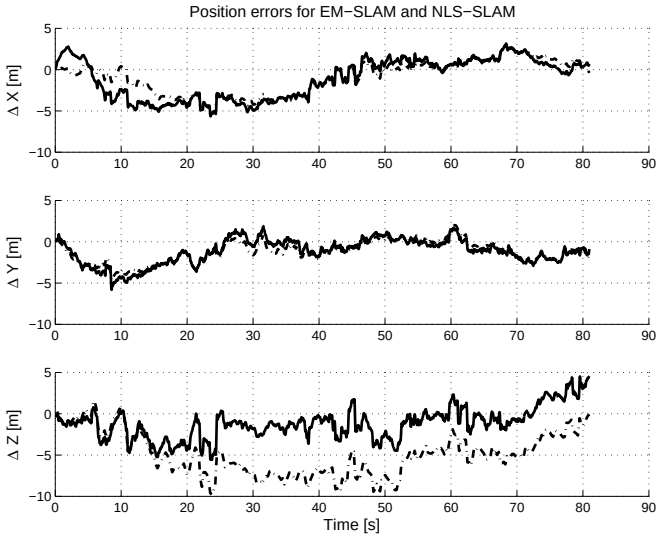


Figure 6: Errors between GPS based and estimated position from EM-SLAM (solid) and NLS-SLAM (dashed).

information about the test platform see e.g., [44].

Camera calibration parameters as well as relative position between the camera and the IMU are known. An open source SIFT implementation from [45] was used to extract the features used as camera measurements. We used a part of the total trajectory, lasting about 80s and making a loop covering about $100 \times 50\ \text{m}^2$, see Figure 5 where the GPS data for the trajectory is shown in the Nort/East local plane. This GPS data is only used as a ground truth in order to evaluate the trajectory estimation accuracy from EM-SLAM and NLS-SLAM methods. In the data, the number of used images is 325 and the number of landmarks is 106.

The position error for each axis is shown in Figure 6 for the EM-SLAM and NLS-SLAM respectively. For the X and Y position, i.e., horizontal plane, both methods have very similar error, of about couple of meters, with maximum of $5\ \text{m}$. For the Z direction, i.e., altitude, EM-SLAM has a smaller error, which is in the same order as horizontal error, while NLS-SLAM has an error which is about twice as big.

In this data set, no ground truth is available for the landmarks, but we evaluate the estimation performance by reprojecting the landmarks 3D position in the images where they are not observed. These reprojections are visually compared with the actual measurements from the images in which these landmarks are observed. Two examples of this comparison are given in Figure 7. Both EM-SLAM and NLS-SLAM give similar performance for map estimation.

IX. Conclusions and Future Work

In this work we present a method for visual/inertial SLAM which uses the EM algorithm to utilize the problem structure and keep low computational complexity for the case where the batch length, N is much larger than the number of landmarks,

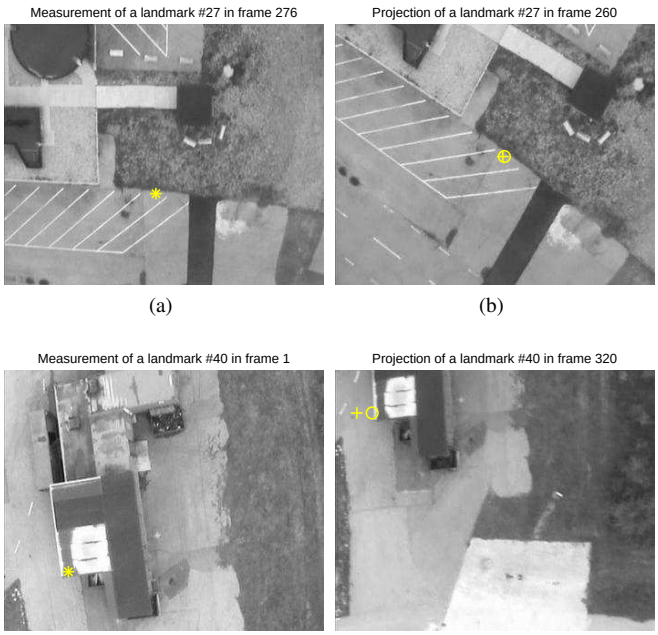


Figure 7: Comparison between measured and reprojected landmarks. Measurement of landmark #27 from image 276 in (a) and its reprojection in image 260 in (b). Measurement of landmark #40 from image 1 in (c) and its reprojection in image 320 in (d). EM-SLAM (○) and NLS-SLAM (+) in both (b) and (d). Note also that landmarks are not measured in the images where they are reprojected.

M while maintaining an accuracy which is comparable to NLS. In average EM-SLAM has $\mathcal{O}(N_c)$ complexity, where N_c is the total number landmark measurements, while NLS-SLAM and GraphSLAM have $\mathcal{O}(N_c N)$. In the worst case, where all landmarks are measured in each frame, both NLS and GraphSLAM have complexity which is quadratic in batch length, while EM-SLAM scales linearly in the batch length. The low computational complexity is obtained by modeling landmarks as static parameters and platform's motion as latent variables.

In future work it would be interesting to change the E-RTS smoother to a particle smoother as it may handle nonlinearities in the models better. Finally, the current implementation is not very efficient in terms of memory or speed and would therefore be interesting to improve and apply to other sensors.

Acknowledgments

Authors would like to thank Mariusz Wzorek and Piotr Rudol at the Department of Computer and Information Science at Linköping University for the Yamaha Rmax data and support with these.

Appendix

Given a smoothed estimate of the latent variables, $\hat{x}^s = \hat{x}_{1:K|K}$ the measurement function $h(x,\theta)$ can be linearised around these as

$$h(x,\theta) \approx \underbrace{h(\hat{x}^s,\theta)}_{\hat{h}} + \underbrace{\nabla_x h(\hat{x}^s,\theta)}_{H}\underbrace{(x-\hat{x}^s)}_{\tilde{x}}. \tag{20}$$

Using this approximation and expanding the norm in (7) for one time instant, while dropping the time index for readability, we obtain

$$\begin{aligned}&\|y-\hat{h}-H\tilde{x}\|^2_{R^{-1}} = (y-\hat{h}-H\tilde{x})^T R^{-1}(y-\hat{h}-H\tilde{x}) = \\ &y^T R^{-1} y - y^T R^{-1}\hat{h} - y^T R^{-1} H\tilde{x} - \\ &\hat{h}^T R^{-1} y + \hat{h}^T R^{-1}\hat{h} + \hat{h}^T R^{-1} H\tilde{x} - \\ &(H\tilde{x})^T R^{-1} y + (H\tilde{x})^T R^{-1}\hat{h} + (H\tilde{x})^T R^{-1}(H\tilde{x})\end{aligned} \tag{21}$$

and taking the expected value

$$\begin{aligned}&\mathbf{E}_{\theta_k}\{\|y-\hat{h}-H\tilde{x}\|^2_{R^{-1}}\} = y^T R^{-1} y - y^T R^{-1}\hat{h} - \\ &\hat{h}^T R^{-1} y + \hat{h}^T R^{-1}\hat{h} + \mathbf{E}_{\theta_k}\{(H\tilde{x})^T R^{-1}(H\tilde{x})|Y\},\end{aligned} \tag{22}$$

since all terms with only $\tilde{x}$ evaluate to zero under the assumption $(\tilde{x}|Y) \sim \mathcal{N}(0, P^s)$. Because $(H\tilde{x})^T R^{-1}(H\tilde{x})$ is scalar it is equal to its trace, and by using the trace rule $\mathrm{Tr}(A^T B A) = \mathrm{Tr}(B A A^T)$ together with the linearity of the trace and expectation operators, the last term becomes

$$\begin{aligned}\mathbf{E}_{\theta_k}\{(H\tilde{x})^T R^{-1}(H\tilde{x})|Y\} &= \mathbf{E}_{\theta_k}\{\mathrm{Tr}((H\tilde{x})^T R^{-1}(H\tilde{x}))|Y\} \\ &= \mathbf{E}_{\theta_k}\{\mathrm{Tr}(R^{-1}H\tilde{x}\tilde{x}^T H^T)|Y\} \\ &= \mathrm{Tr}(R^{-1}H\mathbf{E}_{\theta_k}\{\tilde{x}\tilde{x}^T|Y\}H^T) \\ &= \mathrm{Tr}(R^{-1}HP^sH^T)\end{aligned} \tag{23}$$

which results in

$$\begin{aligned}\mathcal{Q}(\theta,\theta_k) \approx \text{const.} - &\frac{1}{2}\sum_{t\in G}\Big(\|y_t-\hat{h}_t\|^2_{R^{-1}} + \mathrm{Tr}(R^{-1}H_t P^s_{t|N} H_t^T)\Big) = \\ \text{const.} - &\frac{1}{2}\sum_{t\in G}\Big(\|y_t - h_t(\hat{x}_{t|N},\theta)\|^2_{R^{-1}} + \\ &\mathrm{Tr}(R^{-1}\nabla_x h_t(\hat{x}_{t|K},\theta)P^s_{t|K}(\nabla_x h_t(\hat{x}_{t|K},\theta))^T)\Big)\end{aligned} \tag{24}$$

which is the expression in (8).

References

[1] S. Williams, V. Indelman, M. Kaess, R. Roberts, J. Leonard, and F. Dellaert, "Concurrent filtering and smoothing: A parallel architecture for real-time navigation and full smoothing," *International Journal of Robotics Research*, vol. 33, pp. 1544–1568, 2014. [Online]. Available: http://ijr.sagepub.com/content/33/12/1544

[2] M. Kaess and A. Ranganathan and F. Dellaert, "iSAM: Incremental Smoothing and Mapping," *IEEE Transansactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, December 2008.

[3] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular slam," in *Robotics: Science and Systems*, Y. Matsuoka, H. F. Durrant-Whyte, and J. Neira, Eds. The MIT Press, 2010.

[4] A. Kim and R. Eustice, "Pose-graph visual slam with geometric model selection for autonomous underwater ship hull inspection," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 1559–1565.

[5] Z. Sjanic, M. A. Skoglund, T. B. Schön, and F. Gustafsson, "A Nonlinear Least-Squares Approach to the SLAM Problem." in *Proceedings of 18th IFAC World Congress*, Milano, Italy, August/Septemeber 2011.

[6] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *AUTONOMOUS ROBOTS*, vol. 4, pp. 333–349, 1997.

[7] S. Thrun and M. Montemerlo, "The GraphSLAM algorithm with applications to large-scale mapping of urban structures," *INTERNATIONAL JOURNAL ON ROBOTICS RESEARCH*, vol. 25, no. 5, pp. 403–430, 2006.

[8] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2011.

[9] E. Eade, P. Fong, and M. Munich, "Monocular graph SLAM with complexity reduction," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, pp. 3017–3024.

[10] K. Konolige and M. Agrawal, "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1066–1077, 2008.

[11] H. Strasdat, J. M. M. Montiel, and A. Davison, "Real-time monocular slam: Why filter?" in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 2657–2664.

[12] E. Olson, "Real-time correlative scan matching," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 4387–4393.

[13] A. W. Fitzgibbon and A. Zisserman, "Automatic Camera Recovery for Closed or Open Image Sequences," in *ECCV (1)*, 1998, pp. 311–326.

[14] C. Taylor, D. Kriegman, and P. Anandan, "Structure and Motion in Two Dimensions from Multiple Images: A Least Squares Approach," in *Proceedings of the IEEE Workshop on Visual Motion*, Princeton, NJ, USA, October 1991, pp. 242 –248.

[15] D. C. Brown, "A solution to the general problem of multiple station analytical stereo triangulation," Technical Report RCA-MTP Data Reduction, Patrick Airforce Base, Florida, Tech. Rep. Technical Report No. 43 (or AFMTC TR 58-8), 1958.

[16] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Vision Algorithms: Theory and Practice*, ser. Lecture Notes in Computer Science, B. Triggs, A. Zisserman, and R. Szeliski, Eds., vol. 1883. Springer-Verlag, 2000, pp. 298–372.

[17] M. Bryson and M. Johnson-Roberson and S. Sukkarieh, "Airborne smoothing and mapping using vision and inertial sensors," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Kobe, Japan: IEEE Press, 2009, pp. 3143–3148.

[18] L. Kneip, M. Chli, and R. Siegwart, "Robust Real-Time Visual Odometry with a Single Camera and an IMU," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 16.1–16.11, http://dx.doi.org/10.5244/C.25.16.

[19] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart, "Closed-Form Solution for Absolute Scale Velocity Determination Combining Inertial Measurements and a Single Feature Correspondence," in *International Conference on Robotics and Automation*, Shanghai, China, May 2011. [Online]. Available: http://hal.inria.fr/hal-00641772

[20] A. Martinelli, "Vision and IMU Data Fusion: Closed-Form Solutions for Attitude, Speed, Absolute Scale, and Bias Determination," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 44 –60, feb. 2012.

[21] T. Lupton and S. Sukkarieh, "Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 61 –76, feb. 2012.

[22] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, May 2013. [Online]. Available: http://dx.doi.org/10.1177/0278364913481251

[23] V. Indelman, A. Melim, and F. Dellaert, "Incremental light bundle adjustment for robotics navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 11/2013 2013. [Online]. Available: http://www.cc.gatech.edu/∼vindelma/Publications/indelman13iros.pdf

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: http://web.mit.edu/6.435/www/Dempster77.pdf

[25] R. I. Hartley and P. Sturm, "Triangulation," *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146 – 157, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314297905476

[26] Rauch, H. E. and Striebel C. T. and Tung F., "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[27] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[28] T. B. Schön, "An Explanation of the Expectation Maximization Algorithm," Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Tech. Rep. LiTH-ISY-R-2915, Aug. 2009.

[29] A. Wills, B. Ninness, and T. Schön, "Estimating State-Space Models in Innovations Form using the Expectation Maximisation Algorithm," in *The 49th IEEE Conference on Decision and Control (CDC)*, Atlanta, USA, Dec. 2010.

[30] Z. Ghahramani and S. T. Roweis, "Learning Nonlinear Dynamical Systems using an EM Algorithm," in *Advances in Neural Information Processing Systems*, vol. 11. MIT Press, 1999, pp. 599–605.

[31] S. Duncan and M. Gyongy, "Using the EM algorithm to estimate the disease parameters for smallpox in 17th century London," in *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, oct. 2006, pp. 3312 –3317.

[32] E. Ozkan, C. Fritsche, and F. Gustafsson, "Online EM algorithm for joint state and mixture measurement noise estimation," in *Information Fusion (FUSION), 2012 15th International Conference on*, july 2012, pp. 1935 –1940.

[33] S. Le Corff, G. Fort, and E. Moulines, "Online Expectation Maximization algorithm to solve the SLAM problem," in *Statistical Signal Processing Workshop (SSP), 2011 IEEE*, june 2011, pp. 225 –228.

[34] W. J. Rugh, *Linear System Theory*. Prentice Hall, Engelwood Cliffs, NJ, 2nd ed., 1996.

[35] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[36] C. Bibby and I. Reid, "Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association," in *In Proceedings of Robotics: Science and Systems*, 2007.

[37] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun, "EM, MCMC, and Chain Flipping for Structure from Motion with Unknown Correspondence," *Machine Learning*, vol. 50, no. 1-2, pp. 45–71, 2003.

[38] Dellaert, F. and Kaess, M., "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006. [Online]. Available: http://dx.doi.org/10.1177/0278364906072768

[39] M. D. Shuster, "Constraint in attitude estimation part II: Unconstrained estimation," *The Journal of the Astronautical Sciences*, vol. 51, no. 1, pp. 75–101, Jan.–Mar. 2003.

[40] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

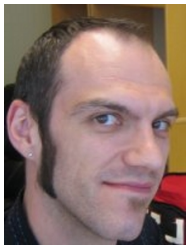
[41] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, T. Kailath, Ed. Prentice-Hall, 1979.

[42] Å. Björck, *Numerical Methods for Least Squares Problems*. SIAM, 1996.

[43] M. A. Skoglund, Z. Sjanic, and F. Gustafsson, "Initialisation and Estimation Methods for Batch Optimisation of Inertial/Visual SLAM," Department of Electrical Engineering, Linköping University, Tech. Rep. LiTH-ISY-R-3065, 2013.

[44] G. Conte and P. Doherty, "Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1–18, Jun. 2009.

[45] A. Vedaldi and B. Fulkerson, "VLFeat: An Open and Portable Library of Computer Vision Algorithms," 2008. [Online]. Available: http://www.vlfeat.org/



Zoran Sjanic received the M.Sc. degree in computer science and engineering 2001 and the Ph.D. degree in Automatic Control in 2013 both from Linköping University. His research interests are sensor fusion for navigation of manned and unmanned aircraft, Simultaneous Localisation and Mapping (SLAM) and nonlinear estimation methods. He is also employed by Saab Aeronautics in Linköping, Sweden, since 2001 where he works at the Sensor Fusion and Tactical Control Department as a system engineer. He also worked as a technical manager for the navigation system in both Gripen fighter aircraft and Skeldar UAV before starting his PhD studies in 2008.

Martin A. Skoglund received the M.Sc. degree in Applied Physics and Electrical Engineering in 2008, and the Ph.D. degree in Automatic Control in 2014, both from Linköping University. At present he is working as a postdoctoral researcher at the Division of Automatic Control at the Department of Electrical Engineering, Linköping University, Linköping, Sweden. His research interests include inertial navigation and mapping, sensor fusion and nonlinear estimation with applications to image processing and mobile robotics.

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 1999-2005 he had a professorship in Communication Systems. His research interests are in stochastic signal processing, adaptive filtering and change detection, with applications to communication, vehicular, airborne, and audio systems. He is a co-founder of the companies NIRA Dynamics (automotive safety systems), Softube (audio effects) and SenionLab (indoor positioning systems).

He was an associate editor for IEEE Transactions of Signal Processing 2000-2006 and is currently associate editor for IEEE Transactions on Aerospace and Electronic Systems and EURASIP Journal on Applied Signal Processing. He was awarded the Arnberg prize by the Royal Swedish Academy of Science (KVA) 2004, elected member of the Royal Academy of Engineering Sciences (IVA) 2007, elevated to IEEE Fellow 2011 and awarded the Harry Rowe Mimno Award 2011 for the tutorial ”Particle Filter Theory and Practice with Positioning Applications”, which was published in the AESS Magazine in July 2010.