Sliding Window Estimation Based on PEM for Visual/Inertial SLAM

ZORAN SJANIC
MARTIN A. SKOGLUND

This paper presents a sliding window estimation method for simultaneous localization and mapping (SLAM) based on the prediction error method (PEM). The estimation problem considers landmarks as parameters while treating dynamics using state space models. The gradient needed for parameter estimation is computed recursively using an extended kalman filter. Results from experiments and simulations with a monocular camera and inertial sensors are presented and compared to batch PEM and nonlinear least-squares SLAM estimators. The presented method maintains good accuracy, and its parametrization is well-suited for online implementation, as it scales better with the size of the problem than batch methods.

Manuscript received October 4, 2023; revised June 11, 2024; released for publication October 14, 2024

Associate Editor handling the review was Ramona Georgescu. The authors gratefully acknowledge funding from the Vinnova Industry Excellence Center LINK-SIC.

Z. Sjanic is with the Department of Automatic Control, 581 83 Linköping University, Linköping, Sweden, and also with the Saab AB, Linköping, Sweden (e-mail: Zoran.Sjanic@liu.se).

M. A. Skoglund is with the Department of Automatic Control, 581 83 Linköping University, Linköping, Sweden, and also with the Eriksholm Research Centre, Oticon A/S, 3070 Snekkersten, Denmark (e-mail: Martin.Skoglund@liu.se).

1557-6418/2024/$17.00 © 2024 JAIF

I. INTRODUCTION

The work in this paper introduces the use of the prediction error method (PEM), see, e.g., [18], as well as its extension to sliding window (SW) as a way to utilize the particular structure of problems encountered in simultaneous localization and mapping (SLAM). The aim in SLAM is to estimate a moving platform's position and orientation while simultaneously mapping the observed environment [1], [7]. A strong trend in SLAM algorithm research is (incremental) batch optimization, which usually solves some form of nonlinear least squares (NLS) problem, see for example [4], [12], [20], [23], [24], [28], [31], [32], [34], [35]. Due to the problem's nature, where both the platform's motion and the environment are considered unknown parameters, solvers are computationally expensive, typically quadratic in the length of the motion parameters; see, e.g., [27]. Throughout the years, many methods were devised aiming at reducing this computational cost. Some of them are utilizing the sparsity of the involved matrices during the calculation in order to use sparse solvers, see, e.g., [10], [11], [20], while others aim at dimensionality reduction of the original problem by solving for only motion parameters while the map is implicit [14], [32] (leading to GraphSLAM), or vice versa by submaps [35]. One issue with the sparsity of the SLAM problem is that it will vary with the problem instance, i.e., the actual environment and the motion. Other methods are utilizing the particular structure of the SLAM problem and aiming at decoupling of the mapping and localization parts, e.g., [26], [27], [29], [33]. These kinds of methods utilizes the intrinsic property of the problem independently of the actual problem instance. Note also that the decoupling must be done in such a way that the correlation between landmarks and the motion is preserved, as it is an inherent property of the SLAM posterior distributions.

The main idea adopted in this work is to model the whole system, i.e., the moving platform and the environment, as a parameterized dynamic model. The environment is represented with discrete points (also called landmarks), which are considered to be the parameters of the system, while the motion of the platform is modeled as a dynamic system. The motivation behind this formulation is that the solution to the problem can be split into two parts, the first part where the landmarks are estimated, and the second part where the motion is estimated. The motion states estimate is here used as a predictor of the system output while utilizing the time series properties through filter solutions. Worth noting is that this property also allows for the gradient of the predictor w.r.t. parameters to be calculated recursively (for a particular choice of the predictor) so that no numerical gradients are necessary. This approach differs from both the standard extended Kalman filter (EKF)-SLAM approach, see, e.g., [7], where both the platform's motion and the landmarks are considered as dynamic states, and the NLS approaches, where everything is considered

as (static) parameters. This division into two parts leads to computation complexity reduction in the following sense:

- The landmark estimation is an optimization problem that is smaller (in the number of estimated parameters) than the standard NLS problem.
- The motion estimation becomes the prediction problem, which is smaller than the usual EKF-SLAM problem since the number of states is constant, i.e., only the motion states are used.

In summary, the optimization problem will scale with the number of parameters (landmarks), while the predictor part will scale with the measurement batch length (time steps). Unlike the approach in [33], the solution proposed here circumvents the need to transform the absolute measurements, i.e., from the platform to the landmarks, to the relative measurements, i.e., between the landmarks. This transformation assumes that range and bearing to the landmarks are measured, and would not work with the original (visual/bearing only) measurements.

The PEM-SLAM batch approach was described in [26], and the extension is to consider an SW adaptation that might enable an efficient online implementation, see e.g., [6], [25]. Online versions of PEM are quite unusual, see, for example, [19], [30], perhaps as excitation and convergence are much harder to obtain and prove. Since the window is just a shorter batch, the SW-PEM-SLAM will consequently be more computationally efficient than SW-NLS-SLAM (given the same window length). This will be shown in an empirical way with Monte Carlo (MC) simulations. The adaptation of PEM-SLAM to an SW is the main contribution of this paper. This adaption is primarily aiming for potential online applications of PEM-SLAM as an estimation method.

The outline of the paper is as follows: In Section II, a brief description of PEM and the model structure is given; In Section III, the expansion of the PEM to the Sliding Window approach is described; In Section IV, the computation complexity of the suggested approach is analyzed; In Section V, the example dynamic and measurements models that are used for evaluation are presented. In Section VI, MC simulation results as well as real data experiments are presented and compared to (SW-)NLS-SLAM, and some empirical computation complexity comparison to (SW-)NLS-SLAM; Section VII ends the paper with conclusions and directions for future work.

II. THE PREDICTION ERROR METHOD

Assume that measurements, $\{y_t\}_{t=1}^N$, from a dynamic system are available. Suppose also that a model of this system is parametrised with some unknown parameters, $\Theta = \{\theta^l\}_{l=1}^L$, and that we want to estimate these using PEM. For that purpose we can use a (quite general) system description of a discrete-time nonlinear state space model as

$$x_{t+1} = f_t(x_t, u_t, w_t, \Theta), \tag{1a}$$

$$y_t = h_t(x_t, \Theta) + e_t. \tag{1b}$$

In this system, the state dynamics is modeled with the function $f_t(\cdot)$, u_t is a known input, w_t is an unknown system noise, the measurement-to-state relation is represented with function $h_t(\cdot)$ and e_t is the measurement noise. The one-step ahead measurement predictor, i.e., predicted measurements at time t given all the information until time $t-1$, is obtained by plugging the predicted state $\hat{x}_{t|t-1}$ into (1b) giving $\hat{y}_{t|t-1}(\Theta) = h_t(\hat{x}_{t|t-1}, \Theta)$. The predictor can, for example, be implemented as an EKF. PEM estimation of Θ is then done by minimizing the sum of the squared norms of the prediction errors, defined as

$$\hat{\Theta} = \arg\min_{\Theta} V(\Theta), \tag{2}$$

and where we have defined $V(\Theta)$ as

$$V(\Theta) = \frac{1}{N}\sum_{t=1}^{N} \mathcal{L}(y_t - \hat{y}_{t|t-1}(\Theta)). \tag{3}$$

$\mathcal{L}$ can be any positive function and $\hat{y}_{t|t-1}(\Theta)$ is defined above. Usually, $\mathcal{L}$ is a standard 2-norm, i.e., $\mathcal{L}(\cdot) = \frac{1}{2}\|\cdot\|_2^2$, leading to the standard (possibly nonlinear) least-squares method. Note that robust norms, such as Huber, can account for spurious data association, see, e.g., [4], but these cases are beyond the scope of this paper.

In this work, we adopt the 2-norm cost function and (3) becomes a standard NLS problem, and any NLS method, such as the Levenberg–Marquardt method [16], [21], can be used to solve it. In this context, it is quite advantageous if the predictor is an analytical function of the parameters and if its gradient w.r.t. to parameters is available. These both qualities simplify and speed up the iterative optimization procedure compared to using numerical methods for calculating the gradient.

A. System Properties and Choice of Predictor

PEM formulations, such as (3), typically result in predictors, $\hat{y}_{t|t-1}(\Theta)$, that are nonlinear in the parameters even for linear Gaussian state space models in (1a). In [17], a recursive PEM method based on the EKF predictor where parameters are appended to the state vector is analyzed. It is shown to be globally asymptotically convergent for the general case (and for linear dynamic systems). This motivates our use of EKF as a predictor in a similar manner to, e.g., [13], [15]. Another advantage is its simple implementation and its possibility to explicitly calculate the gradient of the loss function in (3).

Furthermore, the predictor, $\hat{y}_{t|t-1}(\Theta)$, that will be used (i.e., pinhole camera projection), see Section V-C, is a convex function in parameters, see, e.g., [2], which is a good property for optimization. Another good PEM

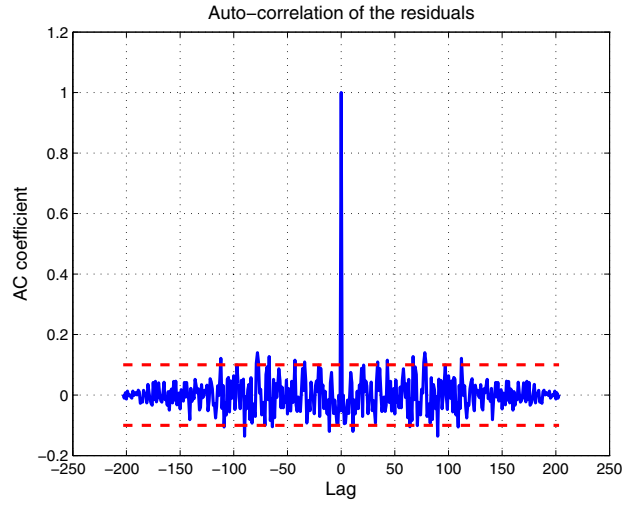

Figure 1. Auto-correlations for the residuals for one landmark and one dimension over the whole trajectory averaged over 50 MC runs. A total of 90% limit is also shown with the red dashed line.

property is that, if the true system is in the model set, if the predictor is stable, and if the innovations are white, then the parameters will converge to true ones; see [18]. In the case for PEM-SLAM here, the model set is modeling the landmarks, which are observed, and consequently it is covering the true environment. The innovations from EKF are (approximately) white, see Fig. 1 where simulation data explained in Section VI-A are used. The residuals' auto-correlations, averaged over 50 MC simulations, are shown for one landmark and in one dimension as an example, but the behavior is very similar for the rest of the landmarks. This fact further motivates the choice of EKF as a predictor. Note that residuals may have heavy-tailed distributions before convergence due to the initialization procedure and also if outliers are not removed.

All the above-mentioned facts support the usage of PEM as a method for solving the SLAM problem. However, with recursive PEM using EKF, the parameters are included in the state vector (with zero dynamics). This is similar to EKF-SLAM, where landmarks are considered as states, [7]. This resemblance highlights the drawbacks of straightforwardly applying PEM to standard EKF-SLAM, which would be severely limited to small maps. Instead, we devise another kind of PEM adaptation, namely SW PEM, which is a main contribution of this paper, and which will be explained in the next section.

III. SLIDING WINDOW PEM ESTIMATION

An adaptation of the PEM is done to accommodate real-time estimation, in a fashion similar to a filter. The adaptation is an iterative procedure where the problem defined in (3) is solved iteratively inside an SW of length K where $K \leq N$. The resulting cost function used in each iteration, i, is then (i is increased by one after each iteration)

$$V_i(\Theta_i) = \frac{1}{2K} \sum_{t=i}^{i+K} \|y_t - \hat{y}_{t|t-1}(\Theta_i)\|_2^2, \tag{4a}$$

$$\hat{\Theta}_i = \arg\min_{\Theta_i} V_i(\Theta_i). \tag{4b}$$

Caution must be taken here when it comes to parameter vector Θ since not all of the parameters might be observed in the window. Hence, Θ_i denotes the subset of the parameters that are observed in the iteration window i. Then, a locally weak observability condition is simply that there are at least as many measurements as parameters in each window and that the resulting observability rank condition [9] is satisfied. The predictor (EKF) in each iteration is initiated by the state and covariance estimate obtained from the previous iteration for the time instance $t = i + 1$ (since this is the first value of the state and covariance in iteration $i + 1$).

A. Statistical Approximation for Information Fusion

To obtain a single best estimate of the static parameters in Θ, we need to fuse different ones from each window. However, double counting of information is a potential problem that can lead to overly confident estimates. This needs to be addressed when parameters are estimated in the moving window. For example, it is very likely that a certain parameter $\theta^l \in \Theta_i$ will be observed and estimated in the window i and $i+1$. In order to handle this (at least approximately), two things are applied:

- The current best parameter estimate is saved.
- The parameter estimates are fused from from the non-overlapping windows only.

The motivation for keeping the current best estimate is that we want to use these in the fusion step, since this will add most information. Also, the estimates from the non-overlapping windows are based on different measurements, which would make them statistically independent. However, this approach is only an approximation since the estimates are dependent on the predictor which, in turn, is dependent on all the past measurements. This dependency will become weaker the further apart in time the state estimates are. Therefore, an assumption is that the estimates from different windows have limited dependency. This is deemed acceptable for our purposes. In the explanation of the fusion principle below, we will omit superindex l in θ^l as notation for an arbitrary parameter in the parameter vector for enhanced readability purposes.

The currently best estimate of the parameter, $\hat{\theta}_b$, is calculated based on an information criterion using the Jacobian $J(\hat{\theta}) = \frac{\partial V(\Theta)}{\partial \theta}\big|_{\theta=\hat{\theta}}$ evaluated with $\hat{\theta}$ from (4b).

Algorithm 1 Sliding Window PEM.

Require: $\hat{x}_{1|1}, P_{1|1}, y_{1:N}, K$
Ensure: $\hat{\Theta}, \hat{x}_{1:N}, P_{1:N}$

for $i = 1 : K - N$
1. Set initial state and covariance for the predictor to $\hat{x}_{i|i}$ and $P_{i|i}$
2. Solve the minimization problem with $V_i(\Theta_i)$ as in (4a)
if $\hat{\theta}^l$ is new parameter **then**
3. Set $\hat{\theta}^l_b$ and $\hat{\theta}^l_f$ to that estimate
else
4. Save the best estimate of the parameters, $\hat{\Theta}_b$, according to (5)
end if
if $\hat{\theta}^l_b$ not updated **and** $\hat{\theta}^l_f$ and $\hat{\theta}^l_b$ are estimated in nonoverlapping windows **then**
5. Fuse $\hat{\theta}^l_f$ and $\hat{\theta}^l_b$ according to (6)
end if
6. Save $\hat{x}_{i|i}$ and $P_{i|i}$
end for
7. $\hat{\Theta} \coloneqq \hat{\Theta}_f$

The following procedure is used:

$$\hat{\theta}_b = \begin{cases} \hat{\theta}_i & \text{if } \mathrm{Tr}(\mathcal{I}_\mathrm{i}) < \mathrm{Tr}(\mathcal{I}_\mathrm{b}). \\ \hat{\theta}_b & \text{otherwise} \end{cases} \tag{5}$$

where index b denominates the window with the best estimate, index i is the current window, and $\mathcal{I} = J(\hat{\theta})^T J(\hat{\theta})$ is the information matrix for the parameter. In this way, we assure that any new estimate will not override the currently best one. We also store the information matrix, $\mathcal{I}_b$ and the window (start and end time indices) together with the current best estimate.

Since we want to utilize all the available information, and get the overall best estimate, fusion from non-overlapping windows is performed. This is done by keeping the fused estimate and the last time index when the fusion was performed. If the parameter's best estimate according to (5) is not updated in the last window, a check is performed to see if the best estimate comes from a window that is not overlapping with the current fused estimate, $\hat{\theta}_f$. In that case, they are fused according to their information representation

$$\hat{\theta}_f = (\mathcal{I}_f + \mathcal{I}_b)^{-1}(\mathcal{I}_f\hat{\theta}_f + \mathcal{I}_b\hat{\theta}_b), \tag{6a}$$

$$\mathcal{I}_f = \mathcal{I}_f + \mathcal{I}_b. \tag{6b}$$

The time index of the fused parameter is then updated accordingly. It shall also be pointed out that the requirement on the nonoverlapping windows is necessary to guarantee that the fused estimates are independent, i.e., estimated with different measurements. The whole approach is summarized as pseudo-code in Algorithm 1.

IV. COMPUTATION COMPLEXITY ANALYSIS

In this section, qualitative computation complexity based on the order of magnitude ($\mathcal{O}$-notation) of the suggested method will be analyzed and compared to primarily NLS method. NLS is chosen since it is *de facto* the standard method and all our numerical results are compared to it. Also, just as in analysis presented in [27], the GraphSLAM method will briefly be mentioned, but no numerical comparison is done. Since this analysis is very similar to the one in the reference above, many details will therefore be omitted here, and the reader is referred to the reference for more information.

The following assumptions and notation will be used: Window size will be denoted by K, and the number of measurements at time t is N_t, and consequently the total number of measurements in the window i is $N_i = \sum_{t=i}^{i+K} \mathsf{N}_t$ (which is linear in the number of time steps in the window, K). The total number of the observed landmarks in the window i is denoted by M_i. Define further the total number of windows as $N_w = N - K + 1$ and the average number of measurements per window as $\bar{N} = \sum_i N_i/N_w$ and the average number of landmarks per window as $\bar{M}$ in the same manner. The average number of landmark measurements per time step is then $\bar{N}^K = \bar{N}/K$, and the average number of measurements per landmark (in a given window) is $\bar{N}^M = \bar{N}/\bar{M}$. We will also assume that the main complexity lies in the calculation of the Jacobian during the iterative optimization procedure, and the analysis will be concentrated on that kind of calculation, i.e., the computation complexity for one iteration step. Notice also that no consideration is taken to any possible sparse structure of the Jacobian and its influence on the solution speed, because that is very much data dependent and will vary between the problem instances.

A. SW-PEM-SLAM

For the SW-PEM-SLAM, the Jacobian of the loss function in (4a) contains only the partial derivatives of the residuals with respect to the landmarks. Since there are, in average, $\bar{N}^M$ measurements per landmark, the total number of operations is proportional to $\bar{N}^M\bar{M}$, i.e., $\mathcal{O}(\bar{N})$. The Jacobian is calculated during the predictor (EKF) run, which has execution complexity proportional to the window length, K, times the complexity of the measurement update, which is in average proportional to the number of measurements updates. This number will be the average number of measurements per time step times the number of time steps, $\bar{N}^K K$, so the complexity is $\mathcal{O}(\bar{N})$, the same as for the Jacobian. All this together gives that the total computation complexity for calculating the Jacobian in the window will be $\mathcal{O}(\bar{N})$. We see that it scales linearly with the size of the window.

B. SW-NLS-SLAM

SW-NLS-SLAM stacks both the landmarks and platform's motion parameters in the parameter vector, which leads to a Jacobian consisting of the partial derivatives of the residuals with respect to both the landmarks and the motion parameters. In our case, there are both acceleration and angular rates motion parameters of size proportional to K in addition to the landmark measurement residuals. The full analysis of NLS-SLAM is done in [27] for the whole batch, and the corresponding result for a window is $\mathcal{O}(K^2 + \bar{N}K + \bar{N} + K)$. Here, we see that the dominating complexity scales quadratically in window size.

C. GraphSLAM

In the GraphSLAM approach, the Jacobian matrix consists of derivatives for positions and rotations with respect to each other, giving a symmetric matrix with dimension proportional to K^2. The analysis of the complexity for GraphSLAM is done in [27], showing that, on average, the complexity will be $\mathcal{O}(\bar{N}^M K)$.

The complexity comparison between SW-PEM-SLAM and SW-NLS-SLAM is illustrated empirically in Section VI.

V. MOTION AND MEASUREMENT MODELS

In this section, the dynamic and sensor measurements models used for estimation in the visual/inartial SLAM problem are introduced. The used sensors are monocular cameras and 6-DOF inertial sensors, i.e., gyroscopes and accelerometers. The camera and inertial sensors are rigidly coupled to the platform in this setup.

The inertial sensors are here treated as inputs to a dynamic system in order to keep the size of the state as small as possible. Also, a minimal 3D point landmark parametrization is used and its measurement function is given by the pinhole projection model.

Throughout the paper, we will use following frames of reference:

- World frame—global frame for expressing platform's and landmarks' position,
- Navigation frame—same as World frame but translated to platform's position,
- Body frame—frame rigidly attached to the platform's body, same origin as Navigation frame but rotated with the body,
- Camera frame—frame rigidly attached to the camera. In general, this can be different than the Body frame, but in this work, we assume that the Camera and Body frames are the same.

All these frames are Cartesian and locally defined. This implies that platforms' and landmarks' position are only estimated locally in the arbitrary chosen World frame. This is customary approach in SLAM since initial position of the platform is usually unknown.

A. State Dynamics

The gyroscope signals, considered as inputs, 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 $u^\mathrm{a} = [u_x^\mathrm{a}, u_y^\mathrm{a}, u_z^\mathrm{a}]^T$. Both of these are measured in the body frame. A discretized dynamic model, where the states are three-dimensional position, velocity, and rotation, $[p_t^T, v_t^T, q_t^T]^T$, in the navigation frame, is then

$$p_{t+1} = p_t + T_s v_t + \frac{T_s^2}{2}\mathrm{R}^T(q_t)(u_t^\mathrm{a} + g^b + w_t^\mathrm{a}), \tag{7a}$$

$$v_{t+1} = v_t + T_s \mathrm{R}^T(q_t)(u_t^\mathrm{a} + g^b + w_t^\mathrm{a}), \tag{7b}$$

$$q_{t+1} = \exp\left(\frac{T_s}{2} S_\omega(u_t^\omega + w_t^\omega)\right) q_t, \tag{7c}$$

where T_s denotes the sampling interval, $\mathrm{R}(q_t)$ is a rotation matrix parametrization 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 is defined as (for each column)

$$\mathrm{R}_{:,1}(q) = \begin{bmatrix} (q^0)^2 + (q^1)^2 - (q^2)^2 - (q^3)^2 \\ 2(q^1 q^2 - q^0 q^3) \\ 2(q^1 q^3 + q^0 q^2) \end{bmatrix}$$

$$\mathrm{R}_{:,2}(q) = \begin{bmatrix} 2(q^1 q^2 + q^0 q^3) \\ (q^0)^2 - (q^1)^2 + (q^2)^2 - (q^3)^2 \\ 2(q^2 q^3 - q^0 q^1) \end{bmatrix}$$

$$\mathrm{R}_{:,3}(q) = \begin{bmatrix} 2(q^1 q^3 - q^0 q^2) \\ 2(q^2 q^3 + q^0 q^1) \\ (q^0)^2 - (q^1)^2 - (q^2)^2 + (q^3)^2 \end{bmatrix}.$$

The gravity expressed in the body frame is $g^b = \mathrm{R}(q_t) g^n$, where $g^n = [0, 0, -g]^T$ is the local gravity vector expressed in the navigation frame, with $g \approx 9.82$, and $\exp(\cdot)$ denotes here the matrix exponential. The noise terms are assumed Gaussian and independent,

$$\left[\left(w_t^\mathrm{a}\right)^T, \left(w_t^\omega\right)^T\right]^T = w_t \sim \mathcal{N}(0, diag(Q_\mathrm{a}, Q_\omega)) = \mathcal{N}(0, Q).$$

For any vector $a = [a_x, a_y, a_z]^T \in \mathbb{R}^3$, the skew-symmetric matrix defined as

$$S_\omega(a) = \begin{bmatrix} 0 & -a_x & -a_y & -a_z \\ a_x & 0 & a_z & -a_y \\ a_y & -a_z & 0 & a_x \\ a_z & a_y & -a_x & 0 \end{bmatrix} \tag{9}$$

is used to parametrize the quaternion dynamics in (7c). It is also worth noticing that, in this case, the dynamics of the system are independent of the landmarks (parameters θ), which simplifies the recursive calculation of the Jacobian.

B. Camera Measurements

The monocular camera is modeled here as a standard pinhole camera, see, cf. [8]. The camera is assumed to be calibrated for its intrinsic parameters (calibration matrix and lens distortion) prior to usage. This enables the usage of the camera as a projective map in Euclidean space, $\mathsf{P} : \mathbb{R}^3 \to \mathbb{R}^2$ by premultiplying the lens undistorted pixel coordinates with the camera calibration matrix. The projection P is defined as

$$\mathsf{P}([X, Y, Z]) = \left[\tfrac{X}{Z}, \tfrac{Y}{Z}\right]^T = [\mathsf{x}_t, \mathsf{y}_t]^T \tag{10}$$

where the Z-coordinate is assumed to be positive and non-zero. A normalized camera measurement, $y_t = [\mathsf{x}_t, \mathsf{y}_t]^T$, of a single landmark, $\theta = [\theta_x, \theta_y, \theta_z]$, at time t is then

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

which relates the three-dimensional position and orientation of the camera to the three-dimensional location of the point. The measurement noise is assumed i.i.d. Gaussian, $e_t = [e_t^{\mathsf{x}}, e_t^{\mathsf{y}}]^T \sim \mathcal{N}(0, R)$.

Equation (11) defines a measurement of one landmark at time t. In order to relate all the measurements to a correct landmark, correspondence variables are used. At time t, correspondence variables $\mathsf{C}_t = \{c_t^j\}_{j=1}^{\mathsf{N}_t} \subseteq \{1, \ldots, M\}$, encode the measurement-landmark assignment, $y_t^j \leftrightarrow \theta^{c_t^j}$. As defined before, N_t is the number of measurements and M is the number of all landmarks. This gives that all observed landmarks at time t are defined as a set $\mathsf{M}_t = \{\theta^{c_t^j}\}_{j=1}^{\mathsf{N}_t}$, where $c_t^j = l$ if a measurement j corresponds to a landmark l. The stacked measurement equation, for all observed landmarks at time t, is then

$$\underbrace{\begin{bmatrix} \mathsf{x}_t^1 \\ \mathsf{y}_t^1 \\ \vdots \\ \mathsf{x}_t^{\mathsf{N}_t} \\ \mathsf{y}_t^{\mathsf{N}_t} \end{bmatrix}}_{y_t^{\text{cam}}} = \underbrace{\begin{bmatrix} \mathsf{P}\left(\mathsf{R}(q_t)(\theta^{c_t^1} - p_t)\right) \\ \vdots \\ \mathsf{P}\left(\mathsf{R}(q_t)(\theta^{c_t^{\mathsf{N}_t}} - p_t)\right) \end{bmatrix}}_{h_t(x_t, \mathsf{M}_t)} + \underbrace{\begin{bmatrix} e_t^{1\mathsf{x}} \\ e_t^{1\mathsf{y}} \\ \vdots \\ e_t^{\mathsf{N}_t\mathsf{x}} \\ e_t^{\mathsf{N}_t\mathsf{y}} \end{bmatrix}}_{e_t^{\text{cam}}}, \tag{12}$$

where $e_t^{\text{cam}} \sim \mathcal{N}(0, R_{\text{cam}})$. R_{cam} is a diagonal matrix since all the measurements are assumed to be mutual independent. Note that Θ_i in (4a) consists of union of the landmarks observed in the window i, i.e., $\Theta_i = \bigcup_{t=i}^{i+K} \mathsf{M}_t$. Note that number of elements in Θ_i is M_i.

Solving the correspondence problem in order to find C_t (also known as data association) is outside the scope for this work and is thus assumed solved.

C. Predictor

The nonlinear predictor needed in PEM is realized with an EKF in our case, since it allows for explicit calculation of the gradient of the loss function in (3). The time update from the EKF produces a predicted state estimate at time t given all the measurements up to time $t-1$, $\hat{x}_{t|t-1}$. This prediction together with the measurement model in (11) can be used to obtain the predicted measurement needed for the PEM loss function, namely $\hat{y}_{t|t-1}(\Theta) = h_t(\hat{x}_{t|t-1}(\Theta), \Theta)$ ($\hat{x}_{t|t-1}(\Theta)$ emphasizes the predicted state's dependency on the parameters). The gradient of the PEM loss function (4a) w.r.t. parameters Θ, needed in the solution procedure, is defined as

$$\begin{aligned} \frac{\partial}{\partial \Theta} V(\Theta) &= \frac{1}{2N} \sum_{t=1}^{N} \frac{\partial}{\partial \Theta} \| y_t - h_t(\hat{x}_{t|t-1}(\Theta), \Theta) \|_2^2 \\ &= \frac{1}{N} \sum_{t=1}^{N} J_t(\Theta)^T r_t(\Theta), \end{aligned} \tag{13}$$

where

$$r_t(\Theta) = y_t - h_t(\hat{x}_{t|t-1}(\Theta), \Theta) \tag{14a}$$

$$J_t(\Theta) = \frac{\partial r_t}{\partial \Theta} = -\frac{\partial h_t(x, \Theta)}{\partial x} \frac{\partial x}{\partial \Theta} - \frac{\partial h_t(x, \Theta)}{\partial \Theta}\bigg|_{x = \hat{x}_{t|t-1}(\Theta)}. \tag{14b}$$

The EKF can calculate the explicit value in (13) during the recursive state and measurement updates. This implies that the computational cost for the gradient is (proportionally) linear in the batch length, and consequently scales better than the NLS, which has (proportionally) quadratic cost, [27]. All the details about how the EKF calculations are done are omitted here, and the reader is referred to [26] instead. With the residual $r_t(\Theta)$ and the Jacobian $J_t(\Theta)$ accessible, any NLS solver can be used to estimate the parameter values. In this particular case, a Levenberg–Marquardt solver is used [16], [21].

VI. RESULTS

The performance of the method is evaluated with MC simulations on the synthetic data, as well as with the real data. We have chosen to evaluate the methods on platforms' and landmarks' positions only. This is due to a lack of ground truth for rotations in the case of real data. All the setup and results are described in the subsections below.

A. Synthetic Data

1) Setup: The simulated trajectory consists of 205 camera measurements at 4 Hz and 2050 acceleration and angular rate measurements at 40 Hz, which gives the total trajectory duration time of 1.24 s. This setup is illustrated in Fig. 2. All simulation examples are based on 50 MC simulations, where the noise on the accelerations, angular rates, and camera measurements is varied for each MC run, and it was sampled from the Gaussian distribution with zero mean and standard deviations $\sigma_a = 10^{-3}\text{m/s}^2$, $\sigma_\omega = 10^{-4} \circ \text{s}^{-1}$, and $\sigma_{\text{cam}} = 10^{-4}\,\text{m}$. As previously explained, the correspondence between

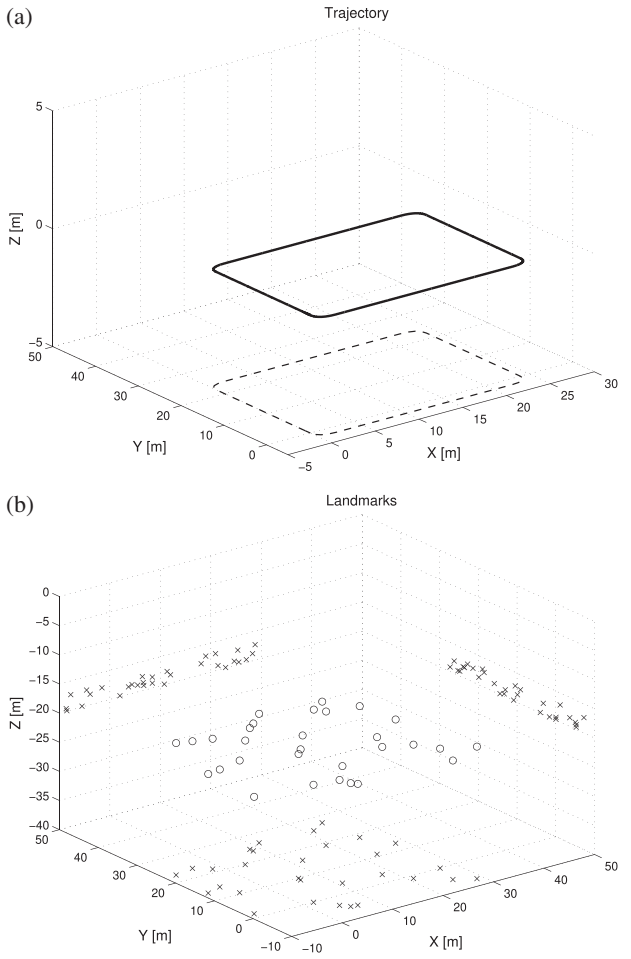

Figure 2. Synthetic environment setup used in MC experiments. (a) Simulated trajectory used in MC simulations with projection on XY-plane for a clearer view (it has constant altitude, i.e., $Z = 0$). (b) Simulated landmarks used in MC simulations with projections on all planes for a clearer view.

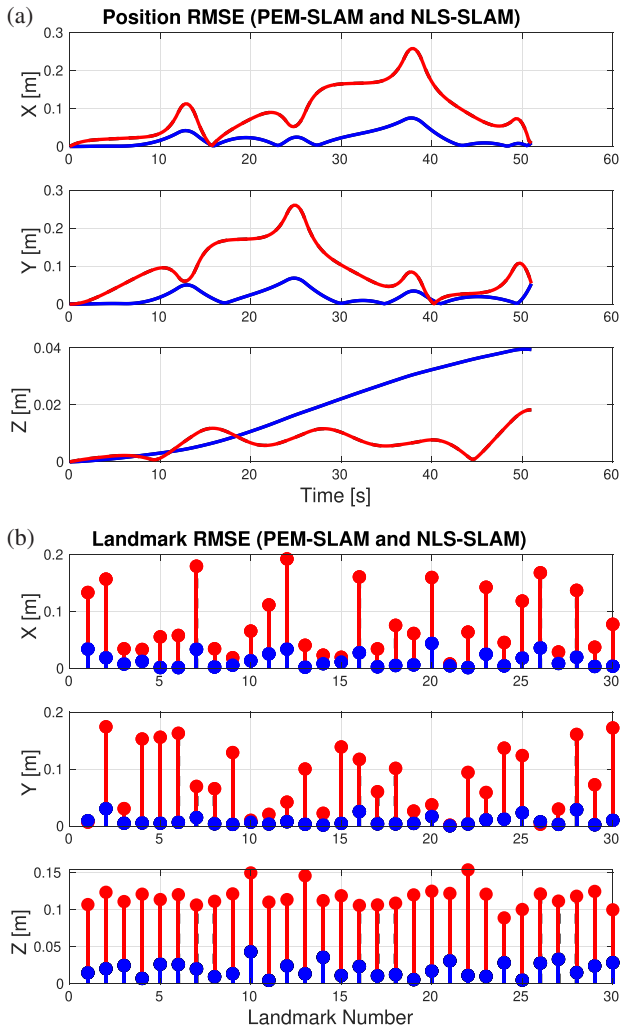

Figure 3. RMS errors for the trajectory and landmarks based on 50 MC simulations for both PEM-SLAM (blue) and NLS-SLAM (red). (a) RMS error for the trajectory estimated with PEM-SLAM (blue) and NLS-SLAM (red) based on 50 MC simulations. (b) RMS error for all the landmarks estimated with PEM-SLAM (blue) and NLS-SLAM (red) for each coordinate based on 50 MC simulations.

measurement and landmarks was known; however, all landmarks were not observed in each camera frame.

2) MC Simulations—Whole batch: The results for the whole batch of data (i.e., "infinite horizon") for both PEM-SLAM and NLS-SLAM are shown in Fig. 3 for the trajectory and the landmarks, respectively.

3) MC Simulations—SW: For the SW approach, 50 MC simulations are done for each of the horizon lengths chosen from {10, 15, 20, 25, 30}. The results, in the form of total position RMSE for the trajectory and some landmarks, for each SW-PEM-SLAM and SW-NLS-SLAM have quite similar and comparable performance, although SW-PEM-SLAM has better RMSE in total for both trajectory and landmarks, just as in the case for the whole batch, see Fig. 4. It is also noticeable that the total error is varying with the horizon length, but not consistently for the SW-PEM-SLAM. It is not necessarily smaller everywhere for the longer horizons, except at the end of the trajectory. SW-NLS-SLAM is more consistent in this regard, at least for the trajectory. Both of the methods are showing this behavior for the landmark position error, i.e., it is not monotonously decreasing with the horizon length for some landmarks, but the general trend is that most landmarks have smaller error for the longer horizon.

B. Real Data

1) Yamaha Rmax: The first real dataset comes from the flight trials performed at Revingehed, Sweden, where a remotely piloted helicopter Yamaha Rmax, see Fig. 5, was flown, [5]. The helicopter was equipped with all the utilized sensors (i.e., IMU and camera). The ground truth (based on GPS) flight trajectory in XY-plane is shown in Fig. 6.

For the validation of the methods, only the horizon length of five images was used, since a short horizon would give a short delay time, which is a realistic assumption for the real-time application. The error between estimated trajectory and GPS- based one for

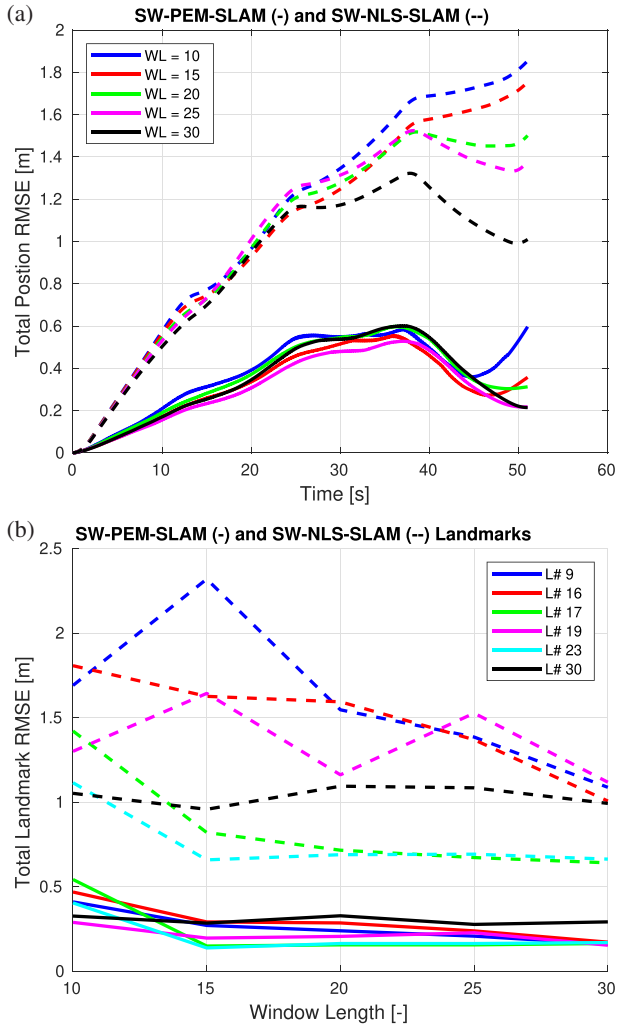

Figure 4. Total RMS errors for the trajectory and seven landmarks for each window length based on 50 MC simulations for SW-PEM-SLAM (solid line) and SW-NLS-SLAM (dashed line). (a) Total RMS error for the trajectory estimated with SW-PEM-SLAM (solid) and SW-NLS-SLAM (dashed) for each window length based on 50 MC simulations. (b) Total RMS error for six landmarks estimated with SW-PEM-SLAM (solid) and SW-NLS-SLAM (dashed) for each window length based on 50 MC simulations.

Figure 5. Remotely piloted helicopter Yamaha Rmax used in the experiments.

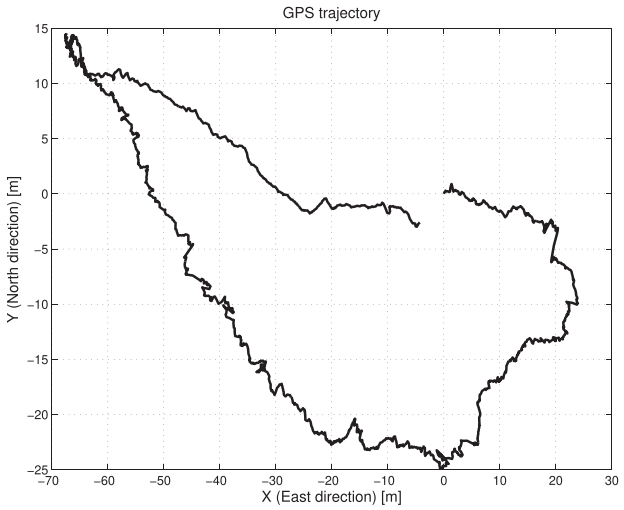

Figure 6. Real data trajectory in XY-plane based on GPS (ground truth).

both SW-PEM-SLAM and SW-NLS-SLAM methods is shown in Fig. 7. Even here, both methods have similar performance for, at least, the X- and Y-coordinates. For the Z-coordinate (or altitude), a much larger error is present. This is a consequence of the inherent visual/inertial SLAM problem property, where it is hard to estimate both distance to landmarks and own localization due to imperfect inertial data. These errors and biases are, in general, not completely removed by the estimation. This can only be done if the inertial data are perfect, which is never the case; see also [22] for further discussion. This behavior is also visible in the simulated data for the Z-coordinate, see, e.g., Fig. 3. In order to remedy this behavior, another kind of stabilizing measurement, like barometric pressure measurements, could be used. Unfortunately, these kinds of measurements were not available. Interestingly, the error is actually decreasing at the end of the trajectory estimated

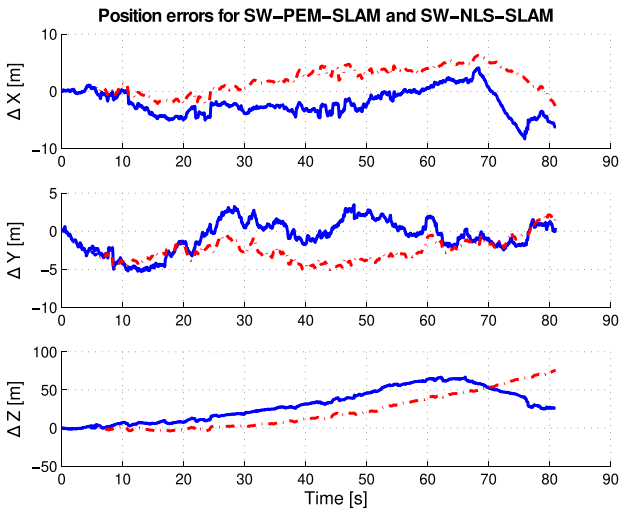

Figure 7. Error between estimated and GPS trajectory for all coordinates for the real data and both SW-PEM-SLAM (blue) and SW-NLS-SLAM (red).

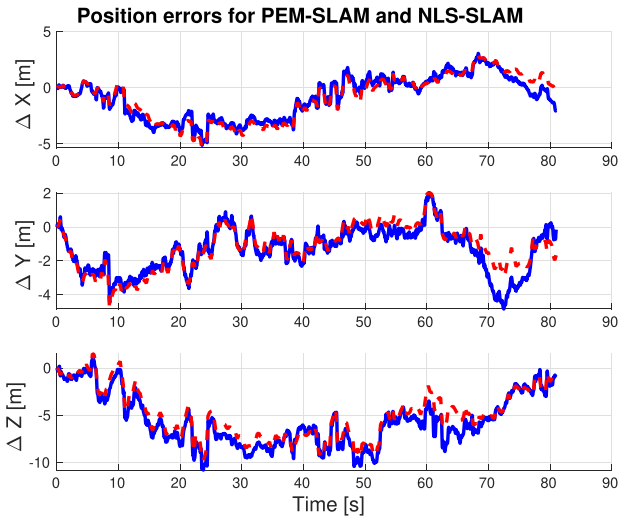

Figure 8. Error between estimated and GPS trajectory for all coordinates for the real data for the whole batch. PEM-SLAM (blue) and NLS-SLAM (red).

by SW-PEM (after 65 seconds approximately). For comparison, the estimate for the whole batch is shown in Fig. 8. Here, the performance is even more similar, and the error in the Z-coordinate is not as prominent (although still larger than X- and Y-coordinates). This is most probably due to the both better loopclosure as well as utilization of the whole data batch instead of only a limited window. Evaluation of map estimation is done by projecting landmarks in an image where they are not observed and comparing these to their measurements from another image where they are observed. This is depicted in Fig. 9. It can be seen that the performance of both methods is quite similar, and that batch estimation has slightly better performance, which is expected.

2) EuRoC: Publicly available datasets from EuRoC MAV [3] have been used in order to evaluate the (SW-) PEM-SLAM performance on another dataset and see how it compares to others' results on the same data. Two out of eleven datasets from EuRoC MAV have been tested. For this dataset, we only show the total RMS error for the whole trajectory and summing all axes as done in [23]. For the SW-PEM-SLAM and dataset MH_01, the RMSE was 0.20 m, and for the PEM-SLAM it was 0.16 m. A horizon of length 20 was used. For the dataset, MH_03 SW-PEM-SLAM had RMSE of 0.32 m with the horizon of length 20. To get the similar performance as for the other dataset, a horizon needed to be 32 long, and the RMSE was 0.21 m in that case. PEM-SLAM had a RMSE of 0.12 m. Compared with the results in, e.g., [23], where RMSE for MH_01 and MH_03 was about 0.07 m, both the SW-PEM-SLAM and PEM-SLAM solutions are in the same order of magnitude. It is worth noting that RMSE in [23] was calculated for keyframes only, while RMSE for SW-PEM-SLAM and PEM-SLAM was for the whole trajectory.

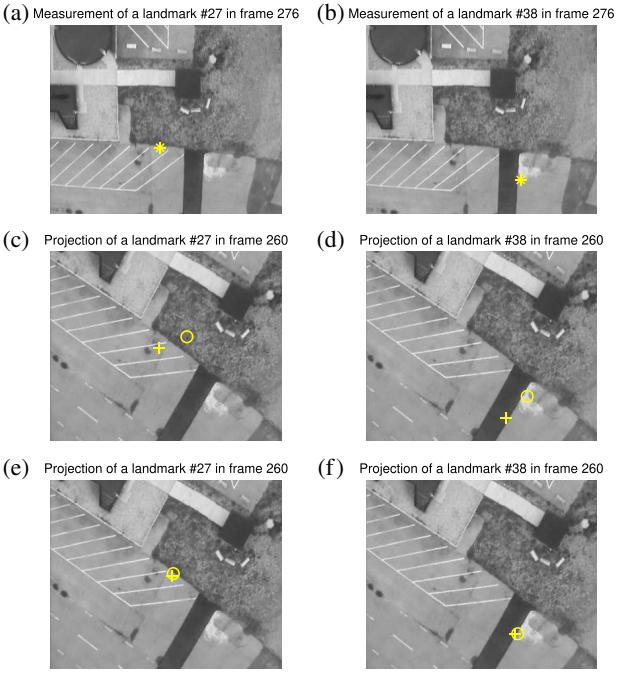

Figure 9. Comparison between measured and reprojected landmarks for the real data and for the SW approach and the whole batch. PEM-SLAM (○) and NLS-SLAM (+) for both SW and batch reprojections. Landmarks are not measured in the images where they are reprojected. (a) Measurement of landmark #27 from image 276. (b) Measurement of landmark #38 from image 276. (c) Reprojection of landmark #27 in image 260 (SW). (d) Reprojection of landmark #38 in image 260 (SW). (e) Reprojection of landmark #27 in image 260 (batch). (f) Reprojection of landmark #38 in image 260 (batch).

C. Execution Time Evaluation

The relative execution time as a function of the horizon length for the SW-PEM-SLAM and the SW-NLS-SLAM are compared. It can be seen in Fig. 10 that the increase for SW-PEM has a linear trend, while for SW-NLS, the trend seems to be quadratic (or at least proportionally quadratic). The plots are produced with the

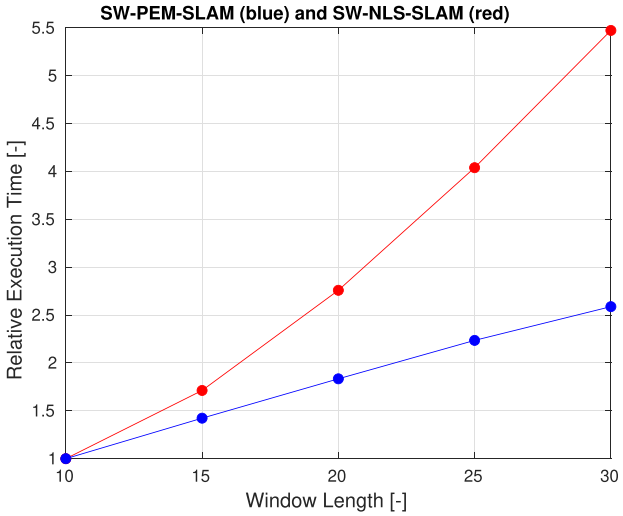

Figure 10. Relative execution time as a function of the horizon length for the SW-PEM-SLAM (blue) and the SW-NLS-SLAM (red).

simulated data averaged over 50 MC runs. Since the estimation accuracy is comparable between the methods, the linear execution time increase for the SW-PEM is a great advantage over SW-NLS when it comes to real-time performance and is one of the main motivations for the choice of SW-PEM-SLAM.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, it is presented how a system identification method, PEM, can be applied to an SLAM problem. This is done by considering the map, here modeled as three-dimensional pointlandmarks, as parameters in the system to be identified, and the motion of the platform observing the landmarks with a monocular camera as dynamic states of the system. Estimation is done in an SW fashion as well as for the whole batch of data. The SW estimation is more appropriate for the real-time adaptation of the estimator, while the whole batch is an offline method. The main advantage of the PEM approach compared to the NLS is the separation between the landmarks and the state estimation, which allows for computation complexity reduction, especially when the horizon length increases. The estimation performance of the SW-PEM-SLAM (and PEM-SLAM, i.e., for the whole batch) is evaluated with MC simulations on both inertial/visual synthetic and real datasets, and compared to SW-NLS-SLAM (and NLS-SLAM) showing comparable performance.

In the continuation of this work, some possible alternative parametrizations of the predictor might be explored, for example, innovation form as in [18]. Also, further development toward better implementation will be pursued.

ACKNOWLEDGMENTS

The authors would like to thank Mariusz Wzorek and Piotr Rudol from AILAB at Linköping University for the Yamaha Rmax data.

REFERENCES

[1] T. Bailey and H. Durrant-Whyte
"Simultaneous localization and mapping (SLAM): Part II,"
IEEE Robot. Automat. Mag., vol. 13, no. 3, pp. 108–117, Sep. 2006.

[2] Stephen Boyd and Lieven Vandenberghe
Convex Optimization. Cambridge, UK: Cambridge University Press, 2004.

[3] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M.W. Achtelik, and R. Siegwart
"The EuRoC micro aerial vehicle datasets,"
Int. J. Robot. Res., vol. 35, no. 10, pp. 1157–1163, 2016.

[4] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós
"ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM,"
IEEE Trans. Robot., vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[5] Patrick Doherty, Patrik Haslum, Fredrik Heintz, Torsten Merz, Per Nyblom, T. Persson, and B. Wingman
"A distributed architecture for autonomous unmanned aerial vehicle experimentation,"
in *Proc. Int. Symp. Distrib. Auton. Robotic Syst.*, 2004, pp. 233–242.

[6] J. Duo, L. Zhao, and J. Mao
"Sliding window based monocular slam using nonlinear optimization,"
in *Proc. 2018 Chin. Intell. Syst. Conf.*, 2019, pp. 519–529.

[7] H. Durrant-Whyte and T. Bailey
"Simultaneous localization and mapping: Part i,"
IEEE Robot.Automat. Mag., vol. 13, no. 12, pp. 99–110, Jun. 2006.

[8] R. I. Hartley and A. Zisserman
Multiple View Geometry in Computer Vision. 2nd ed. Cambridge, U.K.: Cambridge University Press, 2004.

[9] R. Hermann and A. Krener
"Nonlinear controllability and observability,"
IEEE Trans. Autom. Control, vol. 22, no. 5, pp. 728–740, Oct. 1977.

[10] V. Ila, L. Polok, M. Solony, and P. Svoboda
"SLAM—a highly efficient and temporally scalable incremental SLAM framework,"
Int. J. Robot. Res., vol. 36, no. 2, pp. 210–230, 2017.

[11] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert
"iSAM2: Incremental smoothing and mapping using the bayes tree,"
Int. J. Robot. Res., vol. 31, no. 2, pp. 216–235, 2012.

[12] Ayoung Kim and R. Eustice
"Pose-graph visual slam with geometric model selection for autonomous underwater ship hull inspection,"
in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1559–1565.

[13] M. Kok and T. B. Schön
"Maximum likelihood calibration of a magnetometer using inertial sensors,"
in *Proc. 19th IFAC World Congr.*, 2014, pp. 92–97.

[14] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard
"G^2o: A general framework for graph optimization,"
in *Proc. IEEE Int. Conf. Robot. Automat.* 2011, pp. 3607–3613.

[15] R. Larsson, Z. Sjanic, M. Enqvist, and L. Ljung
"Direct prediction-error identification of unstable nonlinear systems applied to flight test data,"
in *Proc. 15th IFAC Symp. Syst. Identification*, 2009, pp. 144–149.

[16] Kenneth Levenberg
"A method for the solution of certain non-linear problems in least squares,"
Quart. J. Appl. Mathmatics, vol. II, no. 2, pp. 164–168, 1944.

[17] L. Ljung
"Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems,"
IEEE Trans. Autom. Control, vol. 24, no. 1, pp. 36–50, Feb. 1979.

[18] L. Ljung
System Identification, Theory for the User, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.

[19] L. Ljung and T. Söderström
Theory and Practice of Recursive Identification, in Signal Processing, Optimization, and Control, vol. 4. Cambridge, MA, USA: MIT Press, 1983.

[20] M. Kaess, A. Ranganathan, and F. Dellaert
"iSAM: Incremental smoothing and mapping,"
IEEE Trans. Robot., vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[21] Donald W. Marquardt
"An algorithm for least-squares estimation of nonlinear parameters,"
SIAM J. Appl. Math., vol. 11, no. 2, pp. 431–441, 1963.

[22] A. Martinelli
"Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Trans. Robot.* vol. 28, no. 1, pp. 44–60, Feb. 2012.

[23] R. Mur-Artal and J. D. Tardós
"ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[24] R. Mur-Artal and J. D. Tardós
"Visual-inertial monocular SLAM with map reuse," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 796–803, Apr. 2017.

[25] G. Sibley, L. Matthies, and G. Sukhatme
Sliding Window Filter for Incremental SLAM. Boston, MA, USA: Springer, 2008, pp. 103–112.

[26] Z. Sjanic and M. A. Skoglund
"Prediction error method estimation for simultaneous localisation and mapping," in *Proc. 19th Int. Conf. Inf. Fusion*, 2016, pp. 927–934.

[27] Z. Sjanic, M. A. Skoglund, and F. Gustafsson
"EM-SLAM with inertial/visual applications," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 1, pp. 273–285, Feb. 2017.

[28] Z. Sjanic, M. A. Skoglund, T. B. Schön, and F. Gustafsson
"A nonlinear least-squares approach to the SLAM problem," in *Proc. 18th IFAC World Congr.* 2011, pp. 4759–4764.

[29] Zoran Sjanic and Martin A. Skoglund
"Exploitation of the conditionally linear structure in visual-inertial estimation," in *Proc. 25th Int. Conf. Inf. Fusion*, 2022, pp. 1–8.

[30] T. Söderström and P. Stoica
System Identification. Hemel Hempstead, UK: Prentice-Hall International, 1989.

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

[32] H. Wang, S. Huang, K. Khosoussi, U. Frese, G. Dissanayake, and B. Liu
"Dimensionality reduction for point feature SLAM problems with spherical covariance matrices," *Automatica*, vol. 51, pp. 149–157, 2015.

[33] Z. Wang, S. Huang, and G. Dissanayake
"D-SLAM: A decoupled solution to simultaneous localization and mapping," *Int. J. Robot. Res.*, vol. 26, no. 2, pp. 187–204, 2007.

[34] 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," *Int. J. Robot. Res.*, vol. 33, pp. 1544–1568, 2014.

[35] L. Zhao, S. Huang, and G. Dissanayake
"Linear SLAM: Linearising the SLAM problems using submap joining," *Automatica*, vol. 100, pp. 231–246, 2019.

Zoran Sjanic (MSc-01, PhD-13) has been employed by Saab Aeronautics, Linköping, Sweden, since 2001, where he is working with the Department of Sensor Fusion and Tactical Control as a System Engineer and Technical Manager for the image analysis and processing system in Gripen fighter aircraft. He has also worked as a Technical Manager for the Navigation System in both Gripen and Skeldar UAS. He also holds a position as an Adjunct Associate Professor in the Division of Automatic Control, Department of Electrical Engineering, at Linköping University, Linköping, Sweden.

His research interests include sensor fusion for navigation of manned and unmanned aircraft, target tracking, simultaneous localization and mapping, and estimation methods for linear and nonlinear systems.

Martin Skoglund (MSc-08, PhD-14) is an Adjunct Associate Professor in the Division of Automatic Control, Department of Electrical Engineering, Linköping University, Linköping, Sweden. Since 2016, he has been working at the Eriksholm Research Centre, Snekkersten, Denmark, part of Oticon, as a Senior Scientist. His research interests include sensor fusion, modeling, and estimation of nonlinear systems. He works with many types of sensors, such as, vision, inertial, audio, EEG, electromagnetic, and more, with the purpose of finding solutions for future hearing care technology.