2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Position Tracking and Sensors Self-Calibration in Autonomous Mobile Robots by Gauss-Newton Optimization

Davide A. Cucci and Matteo Matteucci[1]

Abstract— **The design and development of the pose tracking system for an autonomous mobile robot and the time consuming calibration of its intrinsic sensor parameters (e.g., displacement, misalignment and iron distortions of an inertial measurement unit) are one of the preliminary requirements of any project involving a mobile robot platform. This paper introduces ROAMFREE, a turn-on-and-go multiple sensors pose tracking and self-calibration framework adaptable to different mobile robot platforms (e.g., Ackerman steering vehicles, quadrotor aerial vehicles, omnidirectional mobile robots). We formulate the sensor fusion problem as a Gauss-Newton optimization on an hyper-graph where nodes represent poses and calibration parameters while edges represent nonlinear measurement constraints. This formulation allows us to solve both online pose tracking and offline sensor self-calibration problems.**

I. INTRODUCTION

Any project involving autonomous robots sees the design and development of the pose tracking subsystem as one of the first and most important milestones [1]. This task is often time consuming and the performance of higher level control and navigation modules are tightly dependent on the localization accuracy. Moreover, to determine an estimate of robot position and attitude with respect to a world fixed reference frame, multiple sensor are usually employed which often require the calibration of their intrinsic (e.g., biases and distortions) and extrinsic (e.g., relative displacements with respect to to the robot frame) parameters. Direct measurement of these quantities is often impractical or even impossible and the calibration task is usually performed by means of ad-hoc, hand-tuned, procedures.

The ROAMFREE project [2] aims at developing a general framework for robust multi-sensor fusion and self-calibration to provide an off-the-shelf 6DoF pose tracking system for autonomous mobile robots independent from the specific platform in use. At the present stage of development, it includes a library of sensors which are handled by the framework out-of-the box, an online tracking module based on Gauss-Newton optimization and an offline calibration suite which allows to estimate the unknown sensor parameters such as displacement and misalignment with respect to the robot odometric center, scale factors, biases, and sensor specific parameters such as magnetometers iron distortion.

This work has been supported by the Italian Ministry of University and Research (MIUR) through the PRIN 2009 grant "ROAMFREE: Robust Odometry Applying Multi-sensor Fusion to Reduce Estimation Errors" and Regione Lombardia grant "SINOPIAE"

[1]D. A. Cucci and M. Matteucci are with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy {cucci,matteucci} at elet.polimi.it

Pose tracking in mobile robots has often been addressed in the literature as a problem of multi-sensor data fusion and solved by means of Bayesian filters such as extended Kalman filters and particle filters [3]. This approach has also been used for online estimation of calibration parameters, such as the systematic and non-systematic components of the odometry error [4], or the GPS latency [5]. An EKF based solution for off-the-shelf tracking and sensor self-calibration has been proposed in [6] to target Micro Aerial Vehicles (MAV). If we restrict to the offline calibration of a single sensor displacement and misalignment, the problem is referred as hand-eye calibration [7]. Although the effectiveness of these approaches has been proven and they are now well established in the literature, researchers are often required to write their own ad-hoc implementations to adapt state-of-the-art techniques to the considered application or platform.

In this paper we present an approach to the problem of 6DoF position tracking and sensor self-calibration based on computing the Maximum-A-Posteriori estimate of the joint probabilities of robot poses, given the sensor readings, which is represented as a factor graph. While this approach is often employed for the solution of the Simultaneous Localization and Mapping problem, its application to inertial navigation has appeared recently in the literature [8][9]. In our work, both tracking and calibration problems are solved minimizing the error functions associated to sensor readings by means of Gauss-Newton optimization. In the online tracking case the optimization is performed on a local time window, as it happens in fixed-lag smoothers, while it is performed globally, over a sufficiently long trajectory, in the offline self-calibration case. In Section II we give a high level description of the framework, next we go more in depth in the techniques employed and we describe the mathematical background that allows to perform the Sensor fusion task. In Section V, VI and VII we discuss benchmarks based on synthetic and real world data, as long as conclusions and further work.

II. FRAMEWORK DESCRIPTION

In the ROAMFREE project we aim at the development of a general pose tracking and sensor calibration framework that is independent from the actual robotic platform and sensors in use. To abstract as much as possible from the nature of the information sources we chose not to work directly with sensor hardware, but with *logical sensors* described in terms of their measurement domains, instead of the physical process, and processing, needed to extract such information. To clarify this point, consider a SLAM system built on the top of wheel encoders and multiple cameras. We call

logical sensor the set of the encoders, the cameras and the SLAM algorithm and we treat it as a *black box* information source providing absolute robot pose estimate with respect to a fixed world frame. Moreover, the wheel encoders, taken alone, build up another logical sensor which measures the transformation between successive poses.

We classify all the useful logical sensors for position and attitude tracking according to the type of measurement they produce: (i) absolute position and/or speed, e.g., GPS or SLAM systems, (ii) linear and angular velocity in sensor frame, e.g, gyroscopes, Pitot tubes or visual odometry, (iii) acceleration in sensor frame, e.g., accelerometers, (iv) vector field in sensor frame, e.g., Earth magnetic field or gravitational acceleration. For each of these categories we develop a general error model which relates the noisy sensor measurement to the current state estimate. These models include calibration parameters and take into account all the common sources of distortion and biases, e.g., kinematic parameters such as wheel radius and baseline, sensor displacement or misalignment with respect to the robot odometric center, non-orthogonality of sensor axes, scale factors, and biases.

In the proposed framework, the end-user describes the logical sensors on the robot platform in use choosing among the supported categories, according to the available hardware and software modules; the output of these sensors is delivered to the tracking/calibration system (e.g., through ROS [10] messages) to both obtain a robust estimate of the robot pose and to calibrate unknown sensor parameters. The ROAMFREE framework performs the sensor-fusion task searching for the configuration of robot positions and attitudes which maximize the likelihood of the sensor readings according to their uncertainty. The same optimization is also applied to perform offline estimation of the unknown sensor calibration parameters, provided that a coarse grained estimate is available, relying only on the sensor measurements themselves (i.e., no further information or ground truth is needed).

The ROAMFREE framework is released under the GNU Lesser General Public License and it is available as a C++/Python library or as a ROS node at[1].

III. GAUSS-NEWTON POSE TRACKING AND SENSOR SELF-CALIBRATION

We formulate the tracking and sensor self-calibration problem in terms of a maximum likelihood optimization on a hyper-graph in which the nodes represent poses and sensor parameters while hyper-edges correspond to measurement constraints: we associate an error function to each hyper-edge measuring how well node values fit the sensor observations and we search for a configuration for poses and sensor parameters which minimizes the negative log-likelihood of the graph given all the measurements. If a coarse initial guess is available, a numerical solution can be found by means of the popular Gauss-Newton (GN) or Levenberg-Marquardt (LM) optimization algorithms [11]. In the following this framework is described in details while error models for the different logical sensors are defined in Section IV.

[1]http://roamfree.dei.polimi.it/

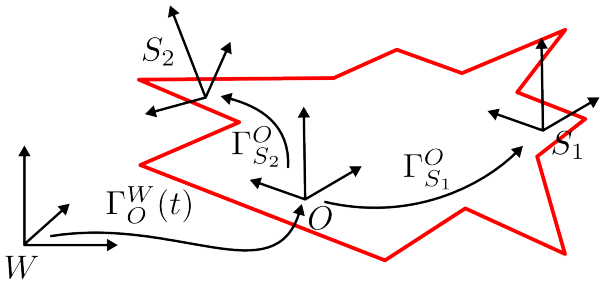


Fig. 1. Reference frames and coordinate transformations in ROAMFREE.

A. Reference Frames

ROAMFREE makes use of three reference frames (see Figure 1): W, the fixed world frame, O, the moving reference frame, placed at the robot odometric center, and S_i, the i-th sensor frame, which is defined with respect to O in terms of it displacement and misalignment calibration parameters.

In the following, a generic reference frame A is represented by the transformation Γ^B_A which expresses the point $p^{(A)}$, whose coordinates are in reference frame A, into the reference frame B. To represent such transformation we keep the coordinates of the origin of A expressed in reference frame B, $A^{(B)}$, and the rotation quaternion q^B_A taking points from frame A to frame B. Thus $\Gamma^B_A = [A^{(B)},\ q^B_A]$.

The tracking module of ROAMFREE estimates position and orientation of O with respect to W, i.e., Γ^W_O. If absolute positioning sensors (e.g., GPS) are not available, at time $t = 0$ we assume that $O \equiv W$ and W becomes arbitrary. Otherwise, the origin of W is placed at given coordinates, with the y axis aligned with the geographic north and the z axis aligned with $\vec{g}$ and pointing up.

B. Problem Definition and Hyper-Graph Construction

The hyper-graph that describes an instance of the tracking and/or sensor self-calibration problem is composed by pose and parameters nodes together with measurement edges (see Figure 2). To construct this graph and determine an initial guess for all the variables in the problem, we start from a high frequency sensor from which it is possible to predict $\Gamma^W_O(t+1)$ given $\Gamma^W_O(t)$ and the sensor measurement $z(t)$. Each time a new reading from this sensor is available, we instantiate a new node $\Gamma^W_O(t+1)$ and we employ the available estimate of the current pose, $\hat{\Gamma}^W_O(t)$, and $z(t)$, to compute the initial guess for the new node. The measurement $z(t)$ is also employed to construct an odometry constraint edge between the pose nodes $\Gamma^W_O(t)$ and $\Gamma^W_O(t+1)$.

When other measurements from different sensors become available, their corresponding edges are inserted into the graph relying on the existing nodes. More precisely, the pose nodes whose timestamp is nearer to the sensor reading one are chosen. For instance, if a new angular velocity reading is available with timestamp t we search in the graph for the pair of subsequent poses nodes with timestamps nearest to t and we insert an angular velocity constraint between them.

Given that all the clocks in the system are synchronized and all the sensor readings are carefully timestamped, this

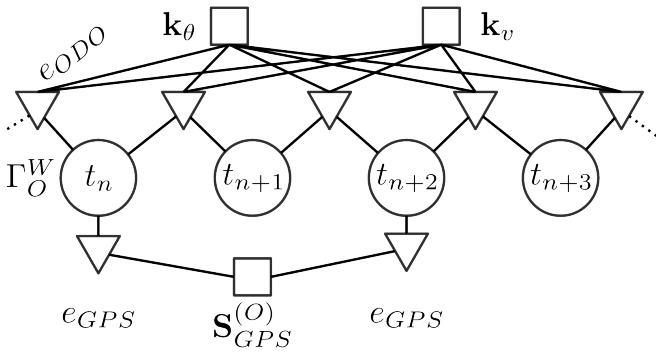


Fig. 2. An instance of the tracking optimization problem with four pose vertices $\Gamma_O^W(t)$ (circles), odometry edges e_{ODO} (triangles), two shared calibration parameters vertices $\mathbf{k}_v$ and $\mathbf{k}_\theta$ (squares), two GPS edges e_{GPS} and the GPS displacement parameter $\mathbf{S}_{GPS}^{(O)}$.

approach natively allows us to deal with out-of-order measurements[2]. Moreover, thanks to the odometry edges being added between every pair of subsequent poses, late measurements still contribute to the most recent pose estimate.

The hyper-graph formulation of the optimization problem is suited to solve both offline sensor calibration and online tracking problems. Indeed, during offline calibration, a set of parameter nodes is chosen for estimation and the graph containing all the available measurements, over a sufficiently long trajectory, is considered. In case of online tracking the parameter nodes are excluded from the optimization: their value is assumed to be known and it is used to compute the errors associated to each edge. To speed up the optimization, we assume the process memory extends up to N time steps in the past: as new nodes are inserted into the graph, the oldest ones are marginalized. Doing so, multiple edges are lost. To prevent information loss, we build a linear constraint that is equivalent, in the neighborhood of the current estimate, with respect to the lost edges and involves the Markov blanket of the marginalized nodes [13]. The optimization is run at constant frequency and we iterate the GN algorithm till timing deadlines approach or when the error function improvement drops below the chosen threshold. Note that hybrid approaches are also possible such as tracking online a limited set of time-varying calibration parameters, along with robot pose, such as gyroscope biases or slippage coefficients.

C. Gauss-Newton Optimization on Manifolds

In position and orientation tracking problems one has often to deal with variables which span over non-Euclidean spaces, such as $SU(2)$ or $SE(3)$. In these cases over-parametrized representations are often employed and the usual formulations of the GN algorithm could fail to preserve the constraints induced by over-parametrization. We address this issue by means of *manifold encapsulation* [14].

Let $e_i(x_i, \eta)$ be the error function associated to the i-th edge in the hyper graph, where x_i is a vector containing the variables appearing in any of the connected nodes and η is a zero-mean Gaussian noise vector. Thus e_i is a random vector and its expected value is approximated as $\overline{e}_i(x_i) = e_i(x_i, \eta)|_{\eta=0}$. Since e_i can involve nonlinear dependencies with respect to the noise, its covariance Σ_η is computed by means of linearization, i.e.:

$$\Sigma_{e_i} = J_{i,\eta} \Sigma_\eta J_{i,\eta}^T \Big|_{x_i = \breve{x}_i, \eta = 0} \tag{1}$$

where $J_{i,\eta}$ is the Jacobian of e_i with respect to η evaluated in $\eta = 0$ and in the current estimate $\breve{x}$.

The maximum likelihood solution for x is given by

$$\mathcal{P}: \quad \arg\min_x \sum_{i=1}^{N} \overline{e}_i(x_i) \Omega_{e_i} \overline{e}_i(x_i) \tag{2}$$

where $\Omega_{e_i} = \Sigma_{e_i}^{-1}$ is the information matrix of the i-th edge. In GN, the error functions are approximated with their first order Taylor expansion around the current estimate $\breve{x}$:

$$e_i(\breve{x} \boxplus \Delta x) \simeq e_i(\breve{x}) + J_{i,\Delta x} \Delta x \tag{3}$$

where $J_{i,\Delta x}$ is the Jacobian of e_i with respect to Δx evaluated at $\Delta x = 0$. Substituting (3) in (2) yields a quadratic form which can be solved in Δx. Then $\breve{x}$ is replaced with $\breve{x} \boxplus \Delta x$ and Ω_{e_i} is updated for each edge as in (1). These steps are repeated till termination criteria are met.

In Equation 3 the x manifold has been *encapsulated* in the sense that the GN algorithm can access the internal representation of x only by means of the $\boxplus$ operator, which, depending on the variable type, consistently maps a local variation Δx in an Euclidean space to a variation on the original manifold. Note that substituting $\boxplus$ with regular $+$ in (3) would yield solutions for Δx which could break the constraints induced by over-parametrization. Consider for instance x being an unit quaternion: a 4D representation is employed to represent rotations in 3D. Here the Δx vector is three-dimensional and the $\boxplus$ operator is defined as

$$x \boxplus \Delta x = x \otimes \begin{bmatrix} 1 - ||\Delta x||, \Delta x_1, \Delta x_2, \Delta x_3 \end{bmatrix} \tag{4}$$

where $\otimes$ is the quaternion product operator and $||\Delta x|| \leq 1$. This ensures that $||x \boxplus \Delta x|| = 1$ and that quaternions are consistently updated once Δx has been determined.

To solve the optimization problem we employ g^{2}o [15], a framework for general graph optimization which relies on very efficient implementations and it is reported to be able to solve graph problems with thousands of nodes in fractions of a second, allowing us to perform the tracking task online.

D. Is the Sensor Self-Calibration Problem Well-Posed?

Here we briefly discuss two issues which can arise in sensor self-calibration problems, i.e. when one seeks to recover calibration parameters without employing an external ground truth but only the noisy sensors readings themselves.

First note that it is not guaranteed that all the calibration parameters are observable given the particular trajectory considered. For instance, the z component of the displacement of an absolute positioning logical sensor, such as a GPS, cannot be estimated if only planar trajectories are considered.

[2]It has been shown in the literature that timing issues could degrade the tracking performances and a number of clock synchronization algorithms, specifically designed for robotics applications, have been proposed [12].

Fortunately, these situations can be detected looking at the approximated Hessian of the error function computed during the Gauss-Newton optimization, from which it is possible to extract an estimate of the parameter marginal covariance. Non-observable parameters would result in high covariances, implying that their variation has limited effect on the error function values. It is difficult to give general theoretical results on the conditions which have to be satisfied by the considered trajectory such that all of a given set of calibration parameters are observable. A detailed analysis for the differential drive robot case can be found in [16].

Moreover, multiple solutions to the calibration problem may exist. Consider for example the case of the magnetometer soft iron distortion matrix and the misalignment with respect to the odometric center (see Section IV-C): as discussed in [17] the magnetometer error model is underdetermined and it is not possible to estimate both parameters at the same time. Again, these situations can be detected looking at the marginal cross-covariance of the parameter variables: high values suggest the existence of dependencies between the involved parameters. In this case one possible strategy is to first obtain a coarse-grained estimate of one set of parameters while keeping the dependent ones fixed.

IV. ERROR MODELS

In this section we will introduce the sensor error models employed in ROAMFREE. We employ bold fonts to highlight calibration parameters. Moreover, since error functions are evaluated multiple times during GN optimization, when we refer to state variables, e.g. $\Gamma_O^W(t)$, we always intend their current estimate $\breve{\Gamma}_O^W(t)$. As a final remark, we use the following notation shortcut: when we have a quaternion left-multiplying a 3D vector we intend that the quaternion has to be replaced with the corresponding 3×3 rotation matrix.

A. Odometer

Every kind of odometer can be seen as a logical sensor which gives an estimation of the robot linear speed and turn rate in sensor frame $\hat{v}^{(S)}(t)$ and $\hat{\omega}^{(S)}(t)$ as noisy functions of the actual sensor reading z_{ODO}, e.g., $\hat{v}^{(S)}(t) = f(z_{ODO}, \eta)$ and $\hat{\omega}^{(S)}(t) = g(z_{ODO}, \eta)$. Here the actual functions f and g can be user defined so that this formulation is suitable to handle raw odometer readings as well as to build a wide variety of logical sensors, ranging from kinematics models with control inputs to visual odometry and scan-matching. We will give an example in Section V.

From $\hat{v}^{(S)}(t)$ and $\hat{\omega}^{(S)}(t)$ we can recover the linear and angular speed estimators for O. For any two reference frames O and S fixed on a rigid body it holds that:

$$\begin{aligned} \hat{v}^{(O)}(t) &= \mathbf{q}_S^O \left[\hat{v}^{(S)}(t) - \hat{\omega}^{(S)}(t) \times \left((\mathbf{q}_S^O)^{-1} \mathbf{S}^{(O)} \right) \right] \\ \hat{\omega}^{(O)}(t) &= \mathbf{q}_S^O \hat{\omega}^{(S)}(t). \end{aligned} \tag{5}$$

Assuming constant speed between t and $t+1$, and for sufficiently small Δt, an accurate approximation of this motion can be obtained by first considering an in-place rotation at $\hat{\omega}^{(O)}(t)$ for $\Delta t/2$, then a translation at $\hat{v}^{(O)}(t)$ for Δt and finally another in-place rotation $\Delta t/2$. Let $\Phi(q, \omega, \Delta t)$ be the function that applies a rotation at turn rate ω for Δt to the orientation represented by the quaternion q:

$$\Phi(q, \omega, \Delta t) = q + \frac{1}{2} q \otimes [0, \omega \Delta t]^T \tag{6}$$

and $q_O'^W(t) = \Phi\left(q_O^W(t), \hat{\omega}^{(O)}(t), \Delta t/2\right)$ be the robot orientation after the first in-place rotation. Note that resulting quaternions in 6 have to be normalized. A prediction for $\Gamma_O^W(t+1)$ reads as

$$\begin{bmatrix} \hat{O}^{(W)}(t+1) \\ \hat{q}_O^W(t+1) \end{bmatrix} = \begin{bmatrix} q_O'^W(t) \hat{v}^{(O)}(t) \Delta t + O^{(W)}(t) \\ \Phi\left(q_O^W(t), \hat{\omega}^{(O)}(t), \Delta t\right) \end{bmatrix}. \tag{7}$$

The error e_{ODO} associated to the odometry edge can be computed as the difference between $\hat{\Gamma}_O^W(t+1)$, which is a noisy function of z_{ODO}, and $\Gamma_O^W(t+1)$:

$$e_{ODO} = \begin{bmatrix} \hat{O}^{(W)}(t+1) - O^{(W)}(t+1) \\ \hat{q}_O^W(t+1) \otimes [q_O^W(t+1)]^{-1} \end{bmatrix}. \tag{8}$$

Note that the Gaussian noise corrupting $\hat{\omega}^{(S)}(t)$, because of the constraints involved in the quaternion representation of 3D rotations, yields a singular covariance matrix for the full quaternion error vector. To overcome this issue, we assume that the orientation error quaternion is near to the identity, i.e. $q_w \approx 1$, and we discard the first component, yielding a full-rank covariance matrix and thus a well defined Ω_{e_i}.

B. GPS

The GPS sensor measures its own latitude, longitude and elevation with respect to a standard, Earth fixed, reference frame. The pose of the robot can be employed to predict the GPS measure, once this has been projected on an Euclidean East, North, Up (ENU) 3D space with origin at W:

$$\hat{z}_{GPS} = O^{(W)} + q_O^W \mathbf{S}^{(O)} + \eta. \tag{9}$$

The same equation can also be employed with other absolute positioning logical sensors, such as SLAM systems. The error associated to the GPS measure is simply $e_{GPS} = \hat{z}_{GPS} - z_{GPS}$ and $e_{GPS} \in \mathbb{R}^3$. Note that taking into account the displacement $\mathbf{S}^{(O)}$ of the GPS sensor with respect to the O reference frame we introduce a correlation between the position measurement and robot orientation.

C. Magnetometer

The three-axis magnetometer reading is a measure of the the Earth's magnetic field $\vec{h}^{(W)}$ in sensor frame. Here we employ an error model which keeps into account hard and soft iron distortion, non-orthogonality of the sensor axis, scaling, bias and misalignment with respect to O [17]. The predicted magnetometer measurement, given the robot orientation at time t, is given by

$$\hat{z}_{MAG}(t) = \mathbf{A} \left(q_O^W(t) \mathbf{q}_S^O \right)^{-1} \vec{h}^{(W)} + \mathbf{b} + \eta, \tag{10}$$

where $\mathbf{A}$ is an unconstrained 3×3 matrix, $\mathbf{b} \in \mathbb{R}^3$ is a bias vector. Finally $e_{MAG} = \hat{z}_{MAG} - z_{MAG}$ and $e_{MAG} \in \mathbb{R}^3$.

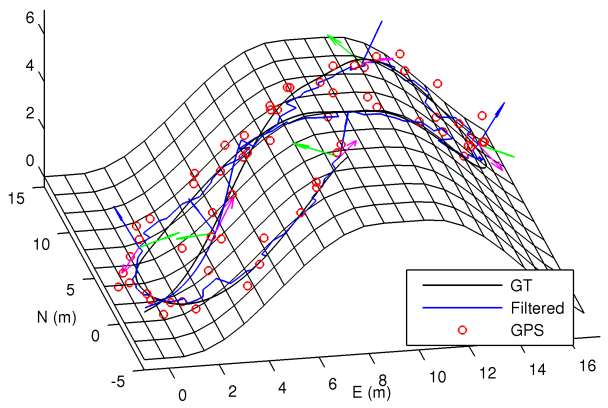


Fig. 3. Ground truth, filtered trajectory and noisy GPS measurement

D. Gyroscope

The gyroscope sensor reads directly $\omega^{(S)}(t)$. As in (7), we compute a prediction of the orientation of the robot at time $t+1$ given $q_O^W(t)$ and the measurement z_{GYRO}:

$$\hat{q}_O^W(t+1) = \Phi\big(q_O^W(t), \mathbf{q}_S^O(z_{GYRO} + \eta_\omega), \Delta t\big). \quad (11)$$

The error associated to the gyroscope measurements is the difference between the predicted orientation $\hat{q}_O^W(t+1)$ and the current estimate $q_O^W(t+1)$:

$$e_{GYRO}(t) = \hat{q}_O^W(t+1) \otimes [q_O^W(t+1)]^{-1}. \quad (12)$$

Again we assume that e_{GYRO} represents a small rotation and we discard the first component of the quaternion.

E. Accelerometer

Accelerometer sensors read the linear acceleration along three axis, including gravity. Here we employ three poses, respectively at times $t-2$, $t-1$ and t, to derive a prediction of the accelerometer reading at time t. We write Equation 7 for the two couples of poses $(\Gamma_O^W(t-2), \Gamma_O^W(t-1))$ and $(\Gamma_O^W(t-1), \Gamma_O^W(t))$. Solving these equations in $\hat{v}^{(O)}$ and $\hat{\omega}^{(O)}$ yields the four estimators $\hat{v}^{(O)}(t-1)$, $\hat{\omega}^{(O)}(t-1)$, $\hat{v}^{(O)}(t)$, $\hat{\omega}^{(O)}(t)$. From the velocities in the O reference frame at time $t-1$ and t we can write an estimator for the linear and angular acceleration of O at time t:

$$\begin{bmatrix} \hat{a}^{(O)}(t) \\ \hat{\alpha}^{(O)}(t) \end{bmatrix} = \frac{1}{\Delta t}\left(\begin{bmatrix} \hat{v}^{(O)}(t) \\ \hat{\omega}^{(O)}(t) \end{bmatrix} - q_{O_{t-1}}^{O_t} \begin{bmatrix} \hat{v}^{(O)}(t-1) \\ \hat{\omega}^{(O)}(t-1) \end{bmatrix} \right), \quad (13)$$

where $q_{O_{t-1}}^{O_t} = (q_O^W(t))^{-1} q_O^W(t-1)$ is the rotation which takes vectors from the O reference frame at time $t-1$ to O at time t. Finally, we predict the linear acceleration at the displaced and misaligned accelerometer as:

$$\begin{aligned} \hat{a}^{(S)}(t) \quad = \quad & (\mathbf{q}_S^O)^{-1}\Big(\big(q_O^W(t)\big)^{-1} \vec{g}^{(W)} + \alpha^{(O)}(t) \times \mathbf{S}^{(O)} + \\ & + \hat{a}^{(O)}(t) + \hat{w}^{(O)}(t) \times \hat{w}^{(O)}(t) \times \mathbf{S}^{(O)} \Big). \end{aligned} \quad (14)$$

and error function becomes $e_{ACC}(t) = \hat{a}^{(S)}(t) - z_{ACC} + \eta$.

V. SIMULATIONS

In this section we discuss sensor self-calibration and tracking experiments for the ROAMFREE framework based on a synthetic dataset. We consider an ATV-like vehicle equipped with an IMU, a GPS and wheel encoders roaming for $30s$ on a hilly terrain and we show that ROAMFREE is able to recover the unknown sensor calibration parameters using only noisy measurement data. Moreover, we run benchmarks for the online tracking module and test the consistency of the estimate with respect to the ground truth.

ATV vehicles are characterized by an Ackermann steering geometry, thus we can construct a logical sensor as in Section IV-A, employing the predictors $\hat{v}^{(S)}(t)$ and $\hat{\omega}^{(S)}(t)$, which are functions of the rear wheel speed and handlebar angle readings, i.e. $z_{ODO}(t) = [z_v(t),\ z_\theta(t)]$. A first order approximation of the motion equations (see [18]) yields:

$$\hat{v}^{(S)}(t) = \begin{bmatrix} \mathbf{k}_v z_v \\ 0 \\ 0 \end{bmatrix},\ \hat{\omega}^{(S)}(t) = \begin{bmatrix} 0 \\ 0 \\ (\frac{\mathbf{k}_v}{\mathbf{L}} z_v) \tan(\mathbf{k}_\theta z_\theta + \mathbf{k}_\gamma) \end{bmatrix}, \quad (15)$$

where $[\mathbf{k}_v, \mathbf{k}_\theta, \mathbf{k}_\gamma]$ is a vector of gains and biases and $\mathbf{L}$ is the distance between front and rear wheel axis.

We place the O reference frame at the middle point of the rear wheel axis and we consider the following set of calibration parameters: $\mathbf{S}_{GPS}^{(O)}$, $\mathbf{q}_{S_{IMU}}^O$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{k}_v/\mathbf{L}$, $\mathbf{k}_\theta$ and $\mathbf{k}_\gamma$ (see Section IV). We compute the sensor readings and then we add Gaussian noise to each component with standard deviations $\sigma_{GPS} = 0.33$, $\sigma_{MAG} = 0.067$, $\sigma_{ACC} = 0.067$, $\sigma_{GYRO} = 0.033$, $\sigma_v = 0.033$ and $\sigma_\theta = 0.014$. From these, we construct the calibration hyper-graph as described in Section III-B initializing the parameter nodes with default values, e.g. $\mathbf{S}_{GPS}^{(O)} = [0,0,0]$ and $\mathbf{A} = I(3)$. To mitigate the issues discussed in Section III-D we divide the calibration parameters in two sets and run the estimation one set at a time: first $[\mathbf{S}_{GPS}^{(O)}, \mathbf{q}_{S_{IMU}}^O]$, then $[\mathbf{A}, \mathbf{b}, \mathbf{k}_s, \mathbf{k}_\theta, \mathbf{k}_\gamma/\mathbf{L}]$. The GN optimization procedure was able to accurately recover the calibration parameters, as shown in Table I.

Next, we run benchmarks for the online tracking module with 40 poses time windows, employing the estimated sensor calibration parameters and considering $N = 50$ different

TABLE I

TRUE VALUES AND ESTIMATES OF THE CALIBRATION PARAMETERS.

	True value			Estimate		
$\mathbf{S}_{GPS}^{(O)}$	0.57			0.645		
	-0.46			-0.341		
	0.32			0.402		
$\mathbf{q}_{S_{IMU}}^O$	0.938			0.945		
	0.104			0.013		
	0.313			0.322		
	0.104			0.005		
$\mathbf{A}$	0.87	-0.02	-0.02	0.87	-0.03	-0.03
	0.00	0.88	0.00	0.01	0.88	0.01
	-0.02	-0.04	0.85	-0.02	-0.05	0.87
$\mathbf{b}$	-0.008			-0.001		
	-0.007			0.003		
	-0.040			-0.033		
$\mathbf{k}_v/\mathbf{L}$	0.8			0.806		
$\mathbf{k}_\theta$	1.0			0.9998		
$\mathbf{k}_\gamma$	0.0			-0.0008		

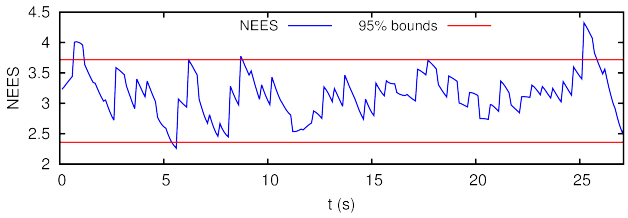


Fig. 4. Average NEES $\bar{\epsilon}(t)$ computed over 50 runs.

realizations of the noise processes. The filtered output for one of these realizations is shown in Figure 3. A measure of the state estimator consistency is found looking at the average Normalized Estimation Error Squared (NEES) over N runs [19, 5.4.2]. Considering the position estimator for the j-th run $\hat{O}_j^{(W)}$, for each time t we have:

$$\epsilon_j = (O^{(W)} - \hat{O}_j^{(W)})^T \left(\Sigma_{\hat{O}^{(W)}}\right)^{-1} (O^{(W)} - \hat{O}_j^{(W)}). \quad (16)$$

Under the hypothesis that the filter is consistent and approximately linear-Gaussian, the average NEES $\bar{\epsilon}$ over N runs is distributed as a χ^2 distribution with $3N$ degrees of freedom. A 95% probability concentration region for $\bar{\epsilon}$ in our case is $\bar{\epsilon} \in [2.359, 3.716]$. If the computed value for $\bar{\epsilon}$ rises over the upper bound, the filter is optimistic, conversely, if it tends below the lower bound, it is conservative. Based on the plot in Figure 4 we can argue the consistency of the robot position estimator, tracked employing the estimated calibration parameters, with respect to the ground truth.

VI. EXPERIMENTAL EVALUATION

In this section we discuss online odometry experiments for two indoor mobile robots, Robocom and Triskar2 (see Figure 5), in which we drive the two robots by means of high amplitude square waves on the linear and angular velocity setpoints, such that relevant wheel slippage occurs. In this situations it becomes difficult to obtain accurate motion estimates employing encoder readings and forward kinematics only. Conversely accurate velocity tracking is achieved thanks to the ROAMFREE sensor fusion engine.

As available sensors, we consider an R2P IMU [20], a Prosilica GC750C camera, and a Hokuyo URG-04LX laser range-finder. Images from the camera feed the *libviso2* monocular visual odometry library [21] and the output is employed to construct a ROAMFREE linear and angular speed logical sensor, while another one handles the output of a 2D scan-matcher working on the Hokuyo data. Two more logical sensor are built as in Equation 15 to handle respectively omnidirectional and differential drive kinematics.

In the considered scenario, no absolute pose feedback is available, thus the robot pose estimate will eventually diverge. Thus we compare the estimated linear and angular velocities with respect to the ground truth provided by an Optitrack motion capture system, and we evaluate the rotational Relative Pose Error [22]: for each couple of poses we compare the relative rotation between them with the corresponding ground truth values. See Figure 6.

The Robocom robot mounts two powerful 150 W Maxon motors which are able to suddenly reverse the wheel speed,

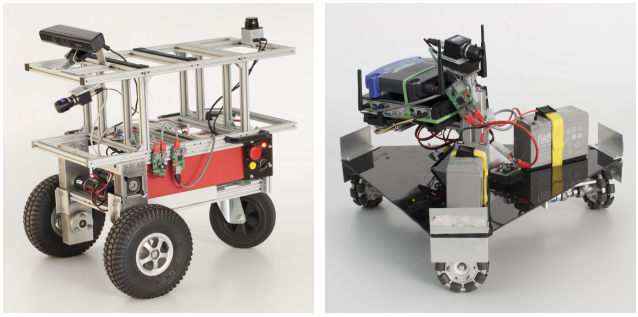


Fig. 5. The Robocom and Triskar2 robots.

incurring in major slippage. Thus, if only the encoder readings were considered, substantial error would be introduced in odometry estimate, as it is possible to see in Figure 6, for instance, at $t = 2.5$ s, where the direct kinematics estimate (blue line) substantially differs from the ground truth (red line) as the tangential speed setpoint is reversed. Conversely, ROAMFREE is able to correct for most of the error on the linear velocity and to track the angular velocity accurately, as it can be seen from the RPE. Similar results are obtained also for the Triskar2 robot, while in this case a further issue consists in the significant vibrations introduced by the omnidirectional wheel rollers. The resulting noise affects gyroscope and accelerometer readings and blurs the Prosilica images, introducing outliers in the inertial and visual odometry measurements. At the present stage of development, outliers can be handled only by means of error functions robustification. Still, it is possible to see that most of the slippage is compensated employing other sensor readings.

VII. DISCUSSION

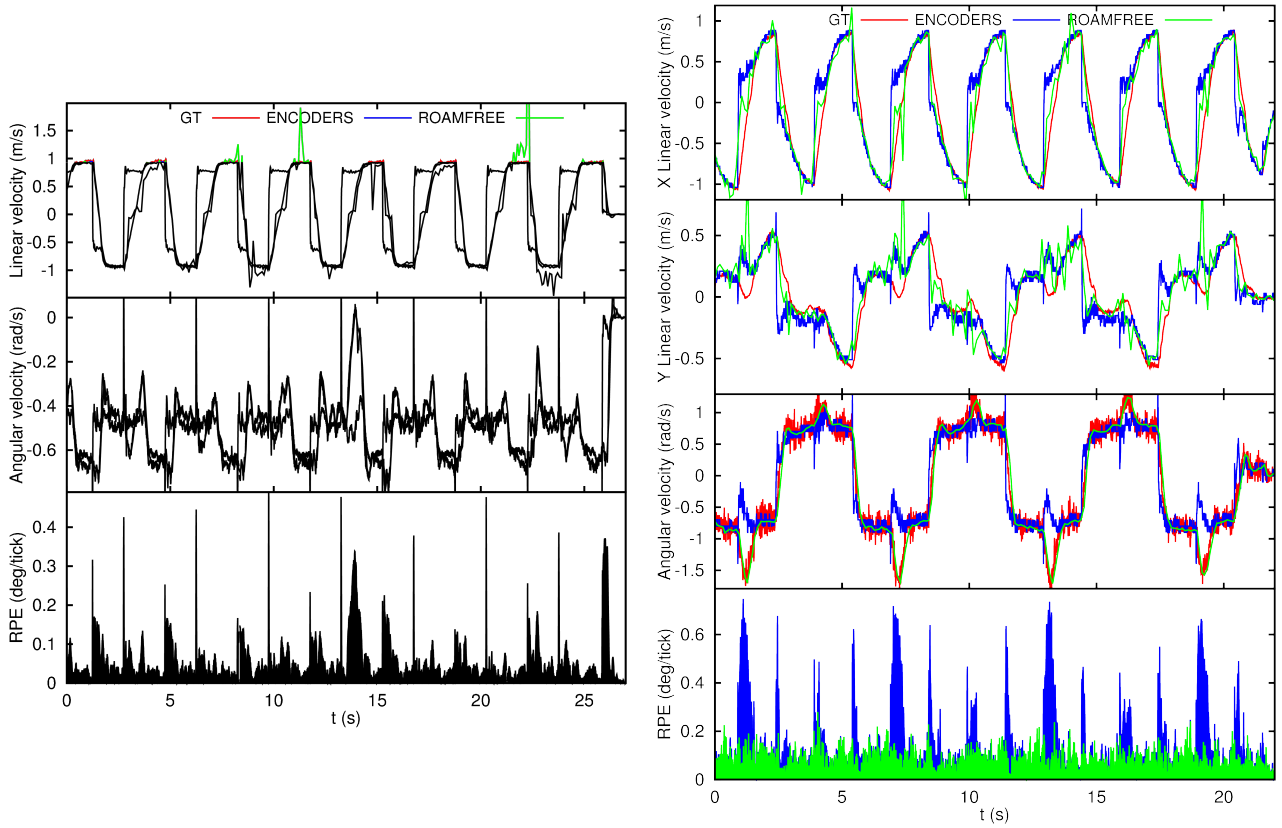
In this work we have presented a flexible and modular software framework which delivers off-the-shelf mobile robot pose tracking and sensor self-calibration capabilities, eventually relieving research from implementing such features from scratch. Moreover, we discussed an experimental evaluation both on simulated and real world data. ROAMFREE undergoes intense development and our ongoing work focuses on the framework structure and improving the mathematical techniques employed to perform the sensor fusion tasks. Moreover, further sensor models are being made available. Among the others, we are working on 3D points reprojection error functions in order to enable the tracking of world fixed features, which would allow to perform SLAM within the ROAMFREE framework, taking advantage of its multi-sensor fusion capabilities, e.g. to handle INS readings.

REFERENCES

[1] J. Borenstein, H. Everett, and L. Feng, "Where am i? sensors and methods for mobile robot positioning," *University of Michigan*, vol. 119, p. 120, 1996.

[2] D. A. Cucci and M. Matteucci, "A flexible framework for mobile robot pose estimation and multi-sensor self-calibration," in *Informatics in Control, Automation and Robotics (ICINCO), 2013 International Conference on*, 2013, pp. 361–368.

[3] Z. Chen, "Bayesian filtering: From kalman filters to particle filters, and beyond," *Statistics*, vol. 182, no. 1, pp. 1–69, 2003.



(a) Robocom (b) Triskar2

Fig. 6. Odometry experiments showing ground truth, direct kinematics and ROAMFREE estimates for linear and angular velocities, along with RPE.

[4] A. Martinelli, N. Tomatis, and R. Siegwart, "Simultaneous localization and odometry self calibration for mobile robot," *Autonomous Robots*, vol. 22, no. 1, pp. 75–85, 2007.

[5] D. Bouvet and G. Garcia, "Improving the accuracy of dynamic localization systems using rtk gps by identifying the gps latency," in *Robotics and Automation (ICRA), 2000 IEEE International Conference on*. IEEE, 2000, pp. 2525–2530.

[6] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous mav," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 31–38.

[7] R. Horaud and F. Dornaika, "Hand-eye calibration," *The international journal of robotics research*, vol. 14, no. 3, pp. 195–210, 1995.

[8] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robotics and Autonomous Systems*, 2013.

[9] R. Kümmerle, G. Grisetti, and W. Burgard, "Simultaneous parameter calibration, localization, and mapping," *Advanced Robotics*, vol. 26, no. 17, pp. 2021–2041, 2012.

[10] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[11] P. E. Gill and W. Murray, "Algorithms for the solution of the nonlinear least-squares problem," *SIAM Journal on Numerical Analysis*, vol. 15, no. 5, pp. 977–992, 1978.

[12] A. Harrison and P. Newman, "Ticsync: Knowing when things happened," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 356–363.

[13] N. Carlevaris-Bianco and R. M. Eustice, "Generic factor-based node marginalization and edge sparsification for pose-graph slam," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5748–5755.

[14] C. Hertzberg, R. Wagner, U. Frese, L. Schrder, and L. Schrder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds." in *Information Fusion*, 2013, pp. 57–77.

[15] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3607–3613.

[16] A. Censi, L. Marchionni, and G. Oriolo, "Simultaneous maximum-likelihood calibration of odometry and sensor parameters," in *Robotics and Automation (ICRA), 2008 IEEE International Conference on*. IEEE, 2008, pp. 2098–2103.

[17] J. Vasconcelos, G. Elkaim, C. Silvestre, P. Oliveira, and B. Cardeira, "Geometric approach to strapdown magnetometer calibration in sensor frame," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 47, no. 2, pp. 1293–1306, 2011.

[18] M. Abe and W. Manning, *Vehicle handling dynamics: theory and application*. Butterworth-Heinemann, 2009.

[19] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. Wiley-Interscience, 2001.

[20] A. Bonarini, M. Matteucci, M. Migliavacca, and D. Rizzi, "R2P: An open source hardware and software modular approach to robot prototyping," *Robotics and Autonomous Systems*, 2013.

[21] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV)*, 2011.

[22] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kummerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tardós, "A comparison of slam algorithms based on a graph of relations," in *Intelligent Robots and Systems (IROS), 2009 IEEE/RSJ International Conference on*. IEEE, 2009, pp. 2089–2095.