Joint on-manifold self-calibration of odometry model and sensor extrinsics using pre-integration

Jérémie Deray[1,2], Joan Solà[1] and Juan Andrade-Cetto[1]

***Abstract*— This paper describes a self-calibration procedure that jointly estimates the extrinsic parameters of an exteroceptive sensor able to observe ego-motion, and the intrinsic parameters of an odometry motion model, consisting of wheel radii and wheel separation. We use iterative nonlinear on-manifold optimization with a graphical representation of the state, and resort to an adaptation of the pre-integration theory, initially developed for the IMU motion sensor, to be applied to the differential drive motion model. For this, we describe the construction of a pre-integrated factor for the differential drive motion model, which includes the motion increment, its covariance, and a first-order approximation of its dependence with the calibration parameters. As the calibration parameters change at each solver iteration, this allows *a posteriori* factor correction without the need of re-integrating the motion data.**

We validate our proposal in simulations and on a real robot and show the convergence of the calibration towards the true values of the parameters. It is then tested online in simulation and is shown to accommodate to variations in the calibration parameters when the vehicle is subject to physical changes such as loading and unloading a freight.

I. Introduction

For a mobile robot, odometry is usually computed composing the motion produced by wheel rotation during a time step to a known pose at a previous step. Since odometry algorithms usually compose small displacements, they have the main drawback of accumulating error unboundedly. Moreover, the computation of a motion increment requires the a priori knowledge of the mobile robot kinematic model parameters. Such parameters are usually estimated using often tedious and complex calibration procedures. A small error in calibration might lead to a bad odometry system and hence to large inaccuracy in the pose estimation over time, preventing other tasks to be performed successfully, such as localization or mapping.

Motion measurements are often reported at very high rates, rendering impracticable to compose individual poses for each measurement. Instead, several measurements are integrated together to a single 'delta' pose increment with respect to a known pose at a previous time step. A change in the integration parameters compels the re-integration of the measurements in order to take into account the parameters correction.

This paper formulates a pre-integrated factor ready to use in any graph-based estimation problem - *e.g.* simultaneous localization and mapping (SLAM) - for the differential drive motion model. Such pre-integrated factors can be corrected *a-posteriori* using pre-computed Jacobians as new estimate of the calibration parameters arrives. This avoids the need to re-integrate all data at each iteration of the optimization. We evaluate it in the context of a joint self-calibration scheme that calibrates both the intrinsic parameters of the differential drive kinematics together with the extrinsic parameters - pose on the robot - of an embedded exteroceptive sensor that can estimate its egomotion. By posing the calibration problem as the optimization of a factor graph and relying on factor pre-integration, the calibration problem can be solved either in batch mode or incrementally online. The online incremental calibration allows to compensate for changes in the kinematics model as they occur, *e.g.*, a heavy load on top of a mobile-base that may slightly squeeze rubber tires.

A. Related work

Most popular odometry calibration procedures rely on the execution by the mobile base of a predetermined trajectory, taking measures of the pose error along the path either manually or with an external sensor. Borenstein and Feng [1] proposed the so called UMBmark test, a calibration procedure requiring the robot to drive along a square path both clockwise and counter-clockwise. Ideally, by the end of the procedure the robot has come back to its initial pose. Measuring and minimizing the pose error between the initial and final pose allows to calibrate the kinematic parameters. Kelly [2] generalized this procedure by replacing the squared shape trajectory by a list of repeatedly visited way-points along a trajectory of any shape.

Martinelli *et al.* [3] proposed to augment the state of a Kalman Filter used for localization with the kinematic parameters. It however requires an a priori known map. Later, the use of an Extended Kalman Filter (EKF) allowed Martinelli *et al.* [4] to simultaneously estimate the systematic and non systematic odometry errors of a mobile robot.

Censi *et al.* [5] proposed the simultaneous calibration of the differential drive kinematics together with the relative 2D pose of a sensor that estimates the robot egomotion. They formulate the calibration problem in terms of a maximum likelihood solved in closed form. Their method does not require any predefined path nor an external sensor but is suited only for parameters that do not vary in time.

[1]The authors are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens Artigas 4-6, 08028, Barcelona, Spain. jderay,jsola,cetto@iri.upc.edu

[2]Jérémie Deray is also with PAL Robotics, Pujades 77-79, 08005 Barcelona, Spain

This work has been supported by the Spanish Ministry of Science, Innovation, and Universities project EB-SLAM (DPI2017-89564-P), by the EU H2020 project LOGIMATIC (H2020-Galileo-2015-1-687534) and by the Spanish State Research Agency through the Maria de Maeztu Seal of Excellence to IRI MDM-2016-0656. J. Deray acknowledges support from the Industrial Doctorate Program of the Catalan Agency for Management of University and Research Grants.

978-1-7281-3605-9/19/$31.00

In a graph-based SLAM context, Kümmerle *et al.* [6] adds the unknown parameters to key-frames states. Doing so allows to consider the kinematic parameters as varying in time (dynamic) if, e.g., the robot was loaded with a cargo heavy enough to alter the previous estimate.

Recently, Cicco *et al.* [7] proposed an unsupervised calibration procedure. By exploring and recording the effects of elementary motions on the uncertainty of the parameters estimate, their method chooses autonomously at every time the best next motion for the robot to perform to further reduce the uncertainty.

The calibration used here can be seen as a middle ground between those of Censi *et al.* [5] and Kümmerle *et al.* [6], in that it allows for the calibration of dynamic kinematic parameters and static sensor extrinsics without a complete SLAM framework around it. We do not require a predefined trajectory, an a-priori known map, nor any external sensor/landmark to perform the calibration.

To approach the calibration of the motion model, we get inspired by the IMU pre-integration theory initiated by [8], later improved in [9], then [10]. We apply it to differential drive motion estimation, and use the mechanisms initially conceived for bias estimation to achieve the self-calibration of the motion model parameters. In this regard, we improve over the formulation in [9] in the sense that we provide recursive integration formulae, and an integration pipeline divided in small steps. This results in equivalent but simpler formulae, especially for the Jacobians through the use of the chain rule. We make systematic use of Lie theory as exposed in [11], making this work a true on-manifold estimation approach.

The remainder of the paper is organized as follows, Section II details the proposed motion pre-integration. Section III explains the joint calibration problem as the optimization of a small factor graph using the pre-integrated factors, with both batch and online methods. Section IV presents experiments using simulations and a real robot and show the convergence of the optimization scheme. Conclusions are reported in Section V.

II. Differential drive pre-integration

A. Abstraction on Lie groups of the pre-integration theory

In a typical mobile robot, motion measurements are acquired at high rate, typically at $100 - 1000$Hz. A measurement $\mathbf{u}_k$ corresponding to a single time increment δt at time t_k produces a local state increment $\delta_k \in \mathcal{M}$, named the 'current delta' through a kinematic motion model characterized by a set of parameters $\mathbf{c}$,

$$\delta_k = \delta(\mathbf{u}_k, \mathbf{c}) \in \mathcal{M} \ . \tag{1}$$

By now, let $\mathcal{M}$ be an abstract Lie group or 'manifold' where the robot state evolves. In fact, measurements typically come in the form of velocities or local increments and can be easily expressed as vectors $\mathbf{b}_k$ in the Lie algebra of $\mathcal{M}$, so that $\delta_k = \mathrm{Exp}(\mathbf{b}_k(\mathbf{u}_k, \mathbf{c}))$. Between two distant time instants t_i and t_k, several δ can be integrated into a single 'delta' or increment $\Delta_{ik} \in \mathcal{M}$ expressing the robot state at time t_k relative to the robot state at t_i (Fig. 1). This integral can be computed recursively through $\Delta_{ik} = \Delta_{ij} \boxplus \delta_k$, where $\boxplus$ indicates the composition law of the Lie group $\mathcal{M}$ and Δ_{ii} is its identity.

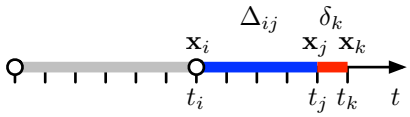


Fig. 1. The pre-integrated delta $\Delta_{ij} \in SE(2)$ contains all motion increments from time i up to time j, so that $\mathbf{x}_j = \mathbf{x}_i \boxplus \Delta_{ij}$. The current delta $\delta_k \in SE(2)$ contains the motion from time j to k, computed from the last motion measurement at time k. We have that $\Delta_{ik} = \Delta_{ij} \boxplus \delta_k$.

The pre-integration theory developed for the IMU sensor [8, 9] deals with the problem of producing motion factors for a factor graph from the aggregation of hundreds of motion measurements. At the time of evaluating the residuals of such factors, one realizes that the integrated IMU delta Δ_{ik} has two undesired dependencies with the state [8]. On one side, it depends on the initial state $\mathbf{x}_i$; on the other side, it depends on the sensor biases $\mathbf{c}$. This can be visualized as

$$\Delta_{ik}(U_{ik}, \mathbf{x}_i, \mathbf{c}) = \Delta_{ij}(U_{ij}, \mathbf{x}_i, \mathbf{c}) \boxplus \delta_k(\mathbf{u}_k, \mathbf{c}) \ , \tag{2}$$

where $U_{ik} = \{\mathbf{u}_i, \cdots, \mathbf{u}_k\}$ is the set of all measurements in the interval. Since the estimates of $\mathbf{x}_i$ and $\mathbf{c}$ change during the optimization, Δ_{ik} would need to be re-integrated at each solver iteration for the residual to be evaluated. This is addressed in two ways. First, a change of reference frame [10] allows us to write a delta that is independent of the initial states. Second, the effect of the change in the calibration $\mathbf{c}$ is linearized around a value $\overline{\mathbf{c}}$ (the 'current' estimate of $\mathbf{c}$ as of the time the linearization is performed) so that the pre-integrated delta can be corrected a posteriori. The result is a pre-integrated delta that only depends on the measured data and $\overline{\mathbf{c}}$,

$$\overline{\Delta}_{ik}(U_{ik}, \overline{\mathbf{c}}) = \overline{\Delta}_{ij}(U_{ij}, \overline{\mathbf{c}}) \boxplus \delta_k(\mathbf{u}_k, \overline{\mathbf{c}}) \ , \tag{3}$$

together with an expression for linearly correcting it when the estimations of bias $\mathbf{c}$ deviate from the values $\overline{\mathbf{c}}$ used during pre-integration,

$$\Delta_{ik}(U_{ik}, \mathbf{c}) \approx \overline{\Delta}_{ik}(U_{ik}, \overline{\mathbf{c}}) \oplus \mathbf{J}_{\mathbf{c}}^{\Delta_{ik}} \cdot (\mathbf{c} - \overline{\mathbf{c}}) \ , \tag{4}$$

where we note $\mathbf{J}_x^y \triangleq \frac{\partial y}{\partial x}$, and $\oplus$ is defined so that $\Delta \oplus \mathbf{v} \triangleq \Delta \boxplus \mathrm{Exp}(\mathbf{v})$ for $\mathbf{v}$ in the Lie algebra of $\mathcal{M}$ (see [11]). This pre-integration avoids the need of re-integrating all motion data at each iteration of the optimizer, as can be seen in the expression of the factor's expectation error,

$$\mathbf{e} = (\overline{\Delta}_{ik}(U_{ik}) \oplus \mathbf{J}_{\mathbf{c}}^{\Delta_{ik}}(\mathbf{c} - \overline{\mathbf{c}})) \ominus (\mathbf{x}_j \boxminus \mathbf{x}_i)) \ , \tag{5}$$

which clearly separates the states $\{\mathbf{x}_i, \mathbf{x}_j, \mathbf{c}\}$ from the measurements U_{ik}. The Jacobian matrix $\mathbf{J}_{\mathbf{c}}^{\Delta_{ik}}$ required for this update is pre-integrated alongside $\overline{\Delta}_{ik}$ during the motion phase. The same is true for the covariances matrix $\mathbf{Q}_{\Delta_{ik}}$ required for computing the residual $\mathbf{r} = \mathbf{Q}_{\Delta_{ik}}^{-\top/2}\mathbf{e}$.

In the following we drop the measurements U_{ij} and $\mathbf{u}_k$ from the notation for simplicity.

B. Differential drive kinematic model

The differential drive model for a ground robot consists of two actuated wheels on a single axle, one on each side of its

base. The robot's frame is defined at the center of the axle, with the X-axis looking forward, *i.e.*, perpendicular to the wheels' axle. The model is parameterized by the wheels radii (r_l, r_r) and the wheel separation d. To each of these values is associated a calibration parameter, $\mathbf{c} = (c_l, c_r, c_d)$, obtaining the calibrated parameters $(c_l r_l, c_r r_r, c_d d)$. The robot state and all deltas lie in $SE(2)$, though we note the orientation part with a simple angle for simplicity.

Motion is measured by means of wheel encoders reporting noisy incremental wheel angles $\mathbf{u} = \delta\boldsymbol{\psi} = (\delta\psi_l, \delta\psi_r)$ every time step. Assuming constant wheel velocities between times t_j and t_k, the motion of the vehicle can be described by a small arc of length δl_k, angle $\delta\theta_k$, and radius $\delta l_k/\delta\theta_k$,

$$\begin{aligned} \delta l_k &= \frac{1}{2}(c_r r_r \delta\psi_{r,k} + c_l r_l \delta\psi_{l,k}) \ , \\ \delta\theta_k &= \frac{1}{c_d d}(c_r r_r \delta\psi_{r,k} - c_l r_l \delta\psi_{l,k}) \ . \end{aligned} \tag{6}$$

This arc can be expressed in the tangent or velocity space of $SE(2)$, *i.e.*, the Lie algebra $se(2)$, with

$$\mathbf{b}_k(\mathbf{u}_k, \mathbf{c}) = [\delta l_k, \delta s_k, \delta\theta_k]^\top \in se(2) \ , \tag{7}$$

where δs_k is a zero-mean perturbation accounting for lateral wheel slippage.

C. Delta pre-integration

Contrary to the IMU case, the 'deltas' of pose in $SE(n)$ are naturally independent of the initial pose $\mathbf{x}_i$. Thus we only need to address the dependency with the calibration parameters $\mathbf{c}$, which we do by setting $\mathbf{b}_k = \mathbf{b}_k(\mathbf{u}_k, \overline{\mathbf{c}})$.

The 'current delta' $\delta_k = (\delta x_k, \delta y_k, \delta\theta_k) \triangleq (\delta\mathbf{p}_k, \delta\theta_k) \in SE(2)$ is computed from the arc (7) using the exponential map $\delta_k = \mathrm{Exp}(\mathbf{b}_k)$ [11] with $\delta s_k = 0$,

$$\begin{aligned} \delta\mathbf{p}_k &= \begin{bmatrix} \frac{\delta l_k}{\delta\theta_k}\sin(\delta\theta_k) \\ \frac{\delta l_k}{\delta\theta_k}(1-\cos(\delta\theta_k)) \end{bmatrix} \approx \begin{bmatrix} \delta l_k \cos(\frac{1}{2}\delta\theta_k) \\ \delta l_k \sin(\frac{1}{2}\delta\theta_k) \end{bmatrix} \\ \delta\theta_k &= \delta\theta_k \ . \end{aligned} \tag{8}$$

where the right-hand expressions account for suitable approximations when $\delta\theta_k \to 0$.

The pre-integrated delta $\overline{\Delta}_{ij} = (\overline{\Delta\mathbf{p}}_{ij}, \overline{\Delta}\theta_{ij}) \in SE(2)$ is the discrete integration of several current deltas δ_k. We implement $\boxplus$ in (3) with the composition law of $SE(2)$,

$$\begin{aligned} \overline{\Delta\mathbf{p}}_{ik} &= \overline{\Delta\mathbf{p}}_{ij} + \overline{\Delta\mathbf{R}}_{ij}\,\delta\mathbf{p}_k \\ \overline{\Delta}\theta_{ik} &= \overline{\Delta}\theta_{ij} + \delta\theta_k \ , \end{aligned} \tag{9}$$

where $\overline{\Delta\mathbf{R}}_{ij} = \mathbf{R}(\overline{\Delta}\theta_{ij}) \triangleq \begin{bmatrix} \cos\overline{\Delta}\theta_{ij} & -\sin\overline{\Delta}\theta_{ij} \\ \sin\overline{\Delta}\theta_{ij} & \cos\overline{\Delta}\theta_{ij} \end{bmatrix}$. Integrated angles are systematically brought back to $(-\pi, \pi]$.

D. Delta Jacobian pre-integration

In this work, all Jacobian and covariance blocks are computed in the Lie-theoretic form, that is, they map vectors in tangent spaces of the involved manifolds — see [11]. They are of dimension 3×3 unless otherwise stated. Their closed form formulae are given in Appendix I.

The Jacobian $\mathbf{J}_{\mathbf{c}}^{\Delta_{ik}}$ in (4) is computed recursively starting at $\mathbf{J}_{\mathbf{c}}^{\Delta_{ii}} = \mathbf{0}$ and using the chain rule,

$$\mathbf{J}_{\mathbf{c}}^{\Delta_{ik}} = \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}} \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}} + \mathbf{J}_{\delta_k}^{\Delta_{ik}} \mathbf{J}_{\mathbf{b}_k}^{\delta_k} \mathbf{J}_{\mathbf{c}}^{\mathbf{b}_k} \ , \tag{10}$$

where $\mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}$, $\mathbf{J}_{\delta_k}^{\Delta_{ik}}$, $\mathbf{J}_{\mathbf{b}_k}^{\delta_k}$ and $\mathbf{J}_{\mathbf{c}}^{\mathbf{b}_k}$ are respectively the Jacobian blocks of (9), (8) and (6–7).

E. Delta covariance pre-integration

Let $\mathbf{Q}_{\boldsymbol{\psi}}$, $\mathbf{Q}_\delta$ and $\mathbf{Q}_\Delta$ be the covariances of respectively the measurement noise, the current delta and the pre-integrated delta. We first define the covariance of the measurement noise,

$$\mathbf{Q}_\psi = \begin{bmatrix} \sigma_{\psi_l}^2 + \alpha^2 & 0 \\ 0 & \sigma_{\psi_r}^2 + \alpha^2 \end{bmatrix} \in \mathbb{R}^{2\times 2} \ , \tag{11}$$

$$\sigma_{\psi_l}^2 = k_l|\delta\psi_l|, \quad \sigma_{\psi_r}^2 = k_r|\delta\psi_r|, \quad \alpha = \tfrac{1}{2}(\mu_l + \mu_r) \ .$$

where k_r and k_l are wheels intrinsics parameters, α acts as an offset equal to half the wheels encoders resolution μ_l and μ_r [12]. The covariance of the current delta δ_k reads,

$$\mathbf{Q}_\delta = \mathbf{J}_{\mathbf{b}_k}^{\delta_k} (\mathbf{J}_{\delta\boldsymbol{\psi}}^{\mathbf{b}_k} \mathbf{Q}_{\boldsymbol{\psi}} \, \mathbf{J}_{\delta\boldsymbol{\psi}}^{\mathbf{b}_k}{}^\top + \mathbf{J}_s^{\mathbf{b}} \sigma_s^2 \mathbf{J}_s^{\mathbf{b}}{}^\top) \mathbf{J}_{\mathbf{b}_k}^{\delta_k}{}^\top \ , \tag{12}$$

with $\mathbf{J}_{\delta\boldsymbol{\psi}}^{\mathbf{b}_k}$, and $\mathbf{J}_s^{\mathbf{b}} = [0, 1, 0]^\top \in \mathbb{R}^3$, the Jacobians of (7), and σ_s^2 the perturbation variance of the wheel slippage δs_k. The motion covariance starts at $\mathbf{Q}_{\Delta_{ii}} = \mathbf{0}$ and is also pre-integrated recursively,

$$\mathbf{Q}_{\Delta_{ik}} = \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}} \mathbf{Q}_{\Delta_{ij}} \, \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}{}^\top + \mathbf{J}_{\delta_k}^{\Delta_{ik}} \mathbf{Q}_\delta \, \mathbf{J}_{\delta_k}^{\Delta_{ik}}{}^\top \ . \tag{13}$$

F. Residual

The residual of the differential drive factors reads,

$$\mathbf{r}(\mathbf{c}) = \boldsymbol{\Omega}^{\top/2}(\Delta(\mathbf{c}) - \widehat{\Delta}) \in \mathbb{R}^3 \ , \tag{14}$$

where $\boldsymbol{\Omega} = \mathbf{Q}_\Delta^{-1}$ is the information matrix of the pre-integrated motion, $\Delta(\mathbf{c})$ comes from (4); and $\widehat{\Delta} = \mathbf{x}_k \boxminus \mathbf{x}_i$ is an independent estimate of the platform motion typically obtained from another embedded sensor (laser scan registration *e.g.* [13], visual odometry, *etc*) an external sensor (Vicon, *etc*), or from a graph with nodes $\mathbf{x}_i, \mathbf{x}_k$.

III. Joint calibration of differential drive intrinsic and sensor extrinsic parameters

We assumed in Section II both $\widehat{\Delta}$ and $\Delta(\mathbf{c})$ to be expressed in the differential drive's reference frame. From now on, we consider $\widehat{\Delta}^S$ to be expressed in another sensor's reference frame, S, which is expressed relative to the robot frame by the extrinsics $\mathbf{T} \triangleq (x, y, \theta) \in SE(2)$. Hereafter we write $\Delta \triangleq \Delta(\mathbf{c})$ for readability.

The aim is now to calibrate jointly the differential drive model $\mathbf{c}$ together with the sensor extrinsics $\mathbf{T}$. From Fig. 2 we clearly see that $\mathbf{T} \boxplus \Delta^S = \Delta \boxplus \mathbf{T}$, so we can define the error of each individual motion delta following this closed kinematic chain

$$\mathbf{e}(\mathbf{c}, \mathbf{T}) = (\mathbf{T} \boxplus \widehat{\Delta}^S) - (\Delta \boxplus \mathbf{T}) \ . \tag{15}$$

A. Jacobians and covariance propagation

The Jacobians of each error $\mathbf{e}_k$ with respect to the unknown $\mathbf{c}$ and $\mathbf{T}$ can be computed by steps, using the Jacobian blocks in Appendix I and the chain rule. Define with Fig. 2,

$$\mathbf{e} = \mathbf{U} - \mathbf{L} \ , \quad \mathbf{U} \triangleq \mathbf{T} \boxplus \widehat{\Delta}^S \ , \quad \mathbf{L} \triangleq \Delta \boxplus \mathbf{T} \ . \tag{16}$$

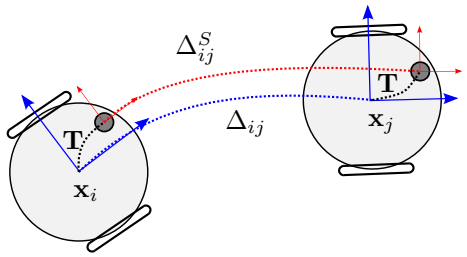


Fig. 2. A differential drive robot moves from pose $\mathbf{x}_i$ to $\mathbf{x}_j$. It mounts an exteroceptive sensor at pose $\mathbf{T}$ (red) with respect to the robot base (blue). It holds that $\Delta_{ij} \boxplus \mathbf{T} = \mathbf{T} \boxplus \Delta_{ij}^S$.

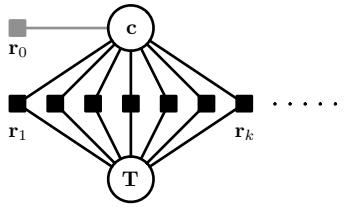


Fig. 3. Factor graph for the estimation problem. Two state blocks $\mathbf{c}$ and $\mathbf{T}$ are linked by a number of factors, each computing a residual of the type $\mathbf{r}_k = \mathbf{\Omega}_k^{\top/2}(\Delta_k(\mathbf{c}) \boxplus \mathbf{T} - \mathbf{T} \boxplus \Delta_k^S)$ (see Fig. 2). An absolute factor (grey) of the type $\mathbf{r}_0 = \mathbf{\Omega}_0^{\top/2}(\mathbf{c} - \mathbf{c}_0)$ keeps $\mathbf{c}$ close to its nominal values $\mathbf{c}_0$.

Then apply the chain rule to find all the Jacobians of $\mathbf{e}$,

$$\mathbf{J}^{\mathbf{e}}_{\Delta} = \mathbf{J}^{\mathbf{e}}_{\mathbf{L}}\mathbf{J}^{\mathbf{L}}_{\Delta} = -\mathbf{J}^{\mathbf{L}}_{\Delta} \quad (17a)$$

$$\mathbf{J}^{\mathbf{e}}_{\widehat{\Delta}^S} = \mathbf{J}^{\mathbf{e}}_{\mathbf{U}}\mathbf{J}^{\mathbf{U}}_{\widehat{\Delta}^S} = \mathbf{J}^{\mathbf{U}}_{\widehat{\Delta}^S} \quad (17b)$$

$$\mathbf{J}^{\mathbf{e}}_{\mathbf{T}} = \mathbf{J}^{\mathbf{e}}_{\mathbf{L}}\mathbf{J}^{\mathbf{L}}_{\mathbf{T}} + \mathbf{J}^{\mathbf{e}}_{\mathbf{U}}\mathbf{J}^{\mathbf{U}}_{\mathbf{T}} = \mathbf{J}^{\mathbf{U}}_{\mathbf{T}} - \mathbf{J}^{\mathbf{L}}_{\mathbf{T}} \quad (17c)$$

$$\mathbf{J}^{\mathbf{e}}_{\mathbf{c}} = \mathbf{J}^{\mathbf{e}}_{\mathbf{L}}\mathbf{J}^{\mathbf{L}}_{\Delta}\mathbf{J}^{\Delta}_{\mathbf{c}} = -\mathbf{J}^{\mathbf{L}}_{\Delta}\mathbf{J}^{\Delta}_{\mathbf{c}} \ . \quad (17d)$$

Then propagate both $\mathbf{Q}_{\widehat{\Delta}^S}$ and $\mathbf{Q}_{\Delta}$ to the space of $\mathbf{e}$,

$$\mathbf{Q}_{\mathbf{e}} = \mathbf{J}^{\mathbf{e}}_{\widehat{\Delta}^S}\mathbf{Q}_{\widehat{\Delta}^S}{\mathbf{J}^{\mathbf{e}}_{\widehat{\Delta}^S}}^{\top} + \mathbf{J}^{\mathbf{e}}_{\Delta}\mathbf{Q}_{\Delta}{\mathbf{J}^{\mathbf{e}}_{\Delta}}^{\top} \quad (18)$$

B. Residual

The residual is similar to (14) with $\mathbf{\Omega} = \mathbf{Q}_{\mathbf{e}}^{-1}$,

$$\mathbf{r} = \mathbf{\Omega}^{\top/2}\,\mathbf{e} \qquad \in \mathbb{R}^3 \quad (19a)$$

$$\mathbf{J}^{\mathbf{r}}_{\mathbf{c}} = \mathbf{\Omega}^{\top/2}\,\mathbf{J}^{\mathbf{e}}_{\mathbf{c}} \ , \quad \mathbf{J}^{\mathbf{r}}_{\mathbf{T}} = \mathbf{\Omega}^{\top/2}\,\mathbf{J}^{\mathbf{e}}_{\mathbf{T}} \quad (19b)$$

C. Batch calibration process

The calibration problem can be modeled as a simple factor graph composed of only two nodes (Fig. 3), one holding the differential drive kinematic parameters $\mathbf{c}$, the second holding the sensor extrinsic parameters $\mathbf{T}$. The nodes are constrained one another by K factors, each containing the pre-integration of a (large) number of motion measurements, whose residuals are evaluated with (19a). Additionally, the $\mathbf{c}$ node is constrained by an absolute factor attracting the calibration parameters towards their nominal values $\mathbf{c}_0$,

$$\mathbf{r}_0(\mathbf{c}) = \mathbf{\Omega}_0^{\top/2}(\mathbf{c} - \mathbf{c}_0) \quad (20)$$

where $\mathbf{\Omega}_0 = \mathrm{diag}(\sigma_l^{-2}, \sigma_r^{-2}, \sigma_d^{-2})$ is chosen sufficiently small not to constrain the optimizer from reaching an adequate solution. In our experiments we chose $\sigma_l = \sigma_r = \sigma_d = 0.01$.

Algorithm 1: Incremental joint self-calibration

Input: $\mathbf{c}_0$, $\mathbf{T}_0$, $\{\Psi_k\}$, $\{\Delta_k^S\}$
$\mathbf{c} = \mathbf{c}_0$, $\mathbf{T} = \mathbf{T}_0$
while *new sensor reading* $\{\Delta_k^S, \mathbf{Q}_k^S\}$ **do**
  $\overline{\mathbf{c}} = \mathbf{c}$
  Pre-integrate diff. drive motion
  $\{\overline{\Delta}, \mathbf{Q}, \mathbf{J}^{\Delta}_{\mathbf{c}}\}_k = integrate(\overline{\mathbf{c}}, \Psi_k)$ (6–13)
  Pack all info for factor k
  $\{\overline{\Delta}, \mathbf{Q}, \mathbf{J}^{\Delta}_{\mathbf{c}}, \overline{\mathbf{c}}, \Delta^S, \mathbf{Q}^S\}_k \rightarrow \Phi_k$
  $\mathbf{x} = (\mathbf{c}, \mathbf{T})$
  while *not end condition* **do**
    $\mathbf{b} = 0, \mathbf{H} = 0$
    for $i \in W$ **do**
      Unpack info for factor i
      $\{\overline{\Delta}, \mathbf{Q}, \mathbf{J}^{\Delta}_{\mathbf{c}}, \overline{\mathbf{c}}, \Delta^S, \mathbf{Q}^S\} \leftarrow \Phi_i$
      $\Delta(\mathbf{c}) = correct(\overline{\Delta}, \mathbf{J}^{\Delta}_{\mathbf{c}}, \mathbf{c}, \overline{\mathbf{c}})$ (4)
      $\{\mathbf{r}, \mathbf{J}^{\mathbf{r}}_{\mathbf{c}}, \mathbf{J}^{\mathbf{r}}_{\mathbf{T}}\}_i = \mathbf{r}_i(\Delta(\mathbf{c}), \mathbf{Q}, \mathbf{T}, \Delta^S, \mathbf{Q}^S)$
      $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{J}_i^{\top}\mathbf{r}_i, \ \mathbf{H} \leftarrow \mathbf{H} + \mathbf{J}_i^{\top}\mathbf{J}_i$ (22)
    Update $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}^{+}\mathbf{b}$ (23–24)
    $(\mathbf{c}, \mathbf{T}) \leftarrow \mathbf{x}$
Output: $\mathbf{c}$,$\mathbf{T}$

Collecting all factors, our estimation problem can be written as

$$[\mathbf{c}^*, \mathbf{T}^*] = \arg\min_{\mathbf{c},\mathbf{T}} \sum_{k=0}^{K} \mathbf{r}_k(\mathbf{c}, \mathbf{T})^{\top}\mathbf{r}_k(\mathbf{c}, \mathbf{T}) \ . \quad (21)$$

A simple solution can be implemented via Gauss-Newton optimization, by iterating until convergence,

$$\mathbf{J}_k = [\mathbf{J}^{\mathbf{r}_k}_{\mathbf{c}} \ \mathbf{J}^{\mathbf{r}_k}_{\mathbf{T}}], \quad \mathbf{b} = \sum_k \mathbf{J}_k^{\top}\mathbf{r}_k, \quad \mathbf{H} = \sum_k \mathbf{J}_k^{\top}\mathbf{J}_k \quad (22)$$

$$\Delta\mathbf{x} = -\mathbf{H}^{+}\mathbf{b} \quad (23)$$

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x} \quad (24)$$

where $\mathbf{x} = (\mathbf{c}, \mathbf{T})$, and $\mathbf{H}^{+}$ is the pseudo-inverse of $\mathbf{H}$.

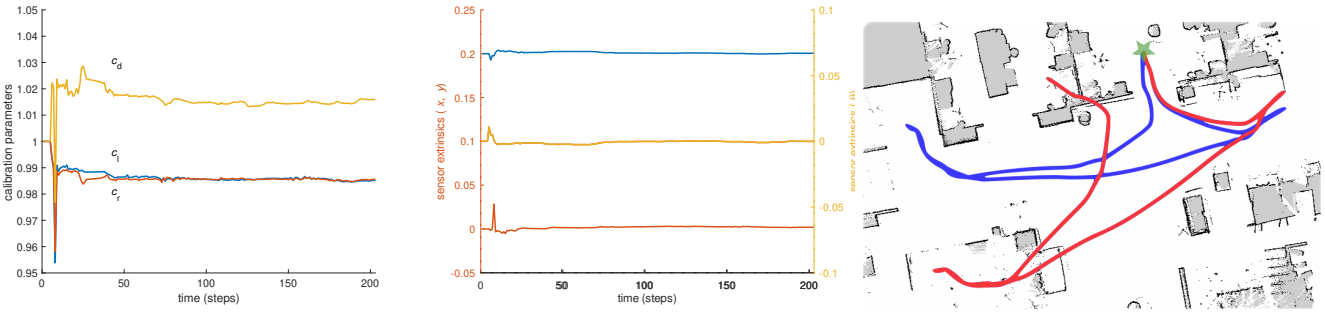
D. Online calibration process

Online incremental self-calibration can be easily achieved by repetitively solving the problem as new factors are incorporated. In order to speed-up operation, the solver is set to escape after a small number of iterations. As factors get old, they gradually accumulate more optimization iterations. Very old factors are removed from the graph, effectively discarding previous information, thus creating a fixed window W of factors being evaluated. The overall algorithm is depicted in Algorithm 1.

One key advantage of the incremental, windowed algorithm is that it allows to deal with dynamic variations of the estimated parameters. Indeed, in case of changes in the parameters to estimate, the different factors cannot reach a good consensus on the states, and the overall cost increases,

$$F(t) = \sum_{i \in W(t)} \mathbf{r}_i^{\top}\mathbf{r}_i \quad (25)$$

By monitoring this cost, we are able to detect these changes, and act on the length of the window W appropriately. A



(a) Evolution of the vehicle kinematics calibration parameters (simulation).

(b) Evolution of the sensor extrinsic parameters (simulation).

(c) Integrated odometry before and after calibration - respectively red and blue (real robot).

Fig. 4. Evolution and effects of the batch calibration.

trivial strategy is to reset the window after a consequent cost increase. Favored strategies simply reduce the window length dynamically. Reducing W has a double effect: on one hand, the factors related to the old values are rapidly removed from the problem, thus canceling their adverse effect; on the other hand, shorter windows allow for faster convergence, at the cost of reduced accuracy. Increasing the window length gradually after observing a recovery in the cost allows to dynamically control the trade-off between convergence speed and accuracy.

IV. Experiments

We evaluate our batch calibration method both on simulated and real data. For the simulation experiment, we use the Robot Operating System [14] simulator Gazebo and the publicly available simulation of the TIAGo[1] robot. The robot is a mobile manipulator, with a differentially driven base equipped with a SICK LMS561 LRF sensor and encoders on each wheel. The real experiment is conducted on a TIAGo-base[2] robot, the mobile-base of the aforementioned TIAGo robot. Finally, the online calibration proposal is evaluated in simulation.

a) Batch calibration experiment: For this experiment, the simulated robot is manually driven along a non-specific path, sufficiently diverse in motions to cover its kinematic space. The motion $(\widehat{\Delta}^S)$ is tracked using a LRF by means of a laser scan matcher algorithm [13]. The precise kinematics parameters are known beforehand, we thus are able to initialize the calibration far from the nominal values and test the viability of the method. From the robot technical specifications we know all parameters,

$$\begin{aligned} r_l = r_r = 0.0985m, d = 0.4044m \\ T_x = 0.202m, T_y = 0m, T_\theta = 0rad \ , \end{aligned} \tag{26}$$

and initialize them to nominal values $\mathbf{c} = [0.1, 0.1, 0.4]$ and $\mathbf{T} = [0.22, 0.1, -0.1]$.

The results are shown in Fig. 4 where the plots show the time evolution of the calibrated parameters. The method is capable of very accurately recovering the vehicle kinematic parameters

$$r_l^* = 0.0985m, \ r_r^* = 0.0986m, \ d^* = 0.4064m \ ,$$

as well as the LRF extrinsics, which converge to

$$T_x^* = 0.2005m, \ T_y^* = 0.0019m, \ T_\theta^* = 0rad \ .$$

A similar setup is used to calibrate the real robot. The nominal values are initialized accordingly to (26). The robot is manually driven along a path (absolute length $\sim 46.5m$) that covers its kinematic space. The motion $(\widehat{\Delta}^S)$ is again tracked using [13]. The calibration results are,

$$\begin{aligned} r_l^* = 0.0986m, \ r_r^* = 0.0978m, \ d^* = 0.4084m \ , \\ T_x^* = 0.1975m, \ T_y^* = 0.0024m, \ T_\theta^* = -0.0108rad \ . \end{aligned}$$

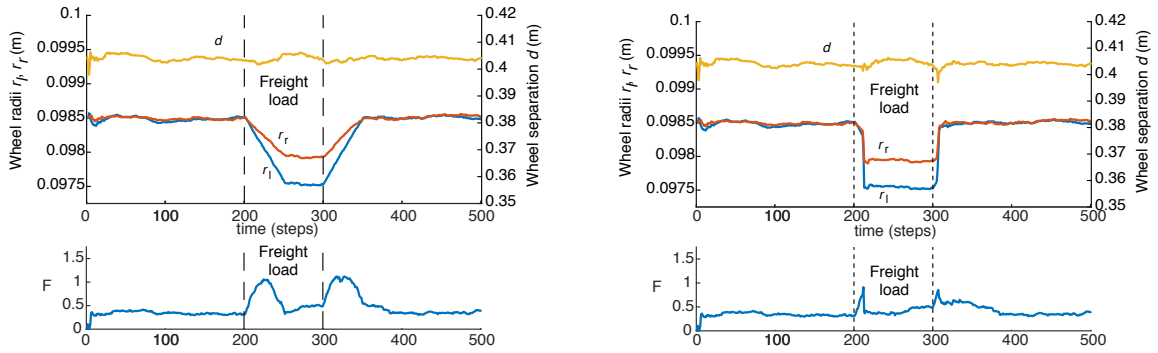
Fig. 4(c) shows the integrated odometry of the robot along a trajectory in an office-like environment before and after calibration. The true initial and final pose of the robot are mingled and mark with a green star in Fig. 4(c). While the trajectory integrated before calibration clearly drifts, with sections of it going through walls and obstacles and its final pose far from the initial one, the trajectory integrated after calibration is much better, with a final pose very close to the initial one.

b) Online calibration experiment: For this experiment, the simulated robot moves along a trajectory. During its course, the wheels radii slightly decrease to simulate the effect of loading and unloading a freight heavy enough to squeeze the rubber tires. Moreover, the freight is not perfectly centered on the robot, leading to an asymmetrical change of the wheels radii. This change in the differential drive model compels the online adaptation of the calibration parameters $\mathbf{c}$ for the reported odometry to remain correct. The simulation runs over 500 time steps; the freight loading and unloading take place respectively at iteration 200 and 300. During this interval, the left and right wheel radii are decreased respectively by 1% and 0.6%, so that

$$r_l = 0.0975m, r_r = 0.0979m, d = 0.4044m \ .$$

Results are reported in Fig. 5 and show an adaptation of the calibration as the freight is loaded then unloaded. The online calibration scheme described in Section III-D is applied both with a fixed size window of 50 factors (Fig. 5(a)) and a dynamically sized window (Fig. 5(b)) triggered by an

[1] http://wiki.ros.org/Robots/TIAGo
[2] http://wiki.ros.org/Robots/TIAGo-base



(a) Window size $= 50$.

(b) Dynamic window size.

Fig. 5. Comparison of the evolution of vehicle kinematic parameters and the aggregated cost factor F for a fixed window size and a dynamic one.

increase in the cost (25). Apart for highlighting the benefit of a quicker transition of the dynamically sized window method, this comparison allows to better visualize the evolution of $\mathbf{c}$ and how it effectively changes toward the true value as older factors escape the window.

V. Conclusions

We have presented a method to jointly self-calibrate the extrinsic parameters of an exteroceptive sensor able to observe ego-motion, and the intrinsic parameters of a differential drive kinematic motion model. An incremental online variant of the method allows to self-calibrate the motion model while it is subject to physical change. We evaluated our proposal in simulation and shown that it converges toward the true values of the parameters. We also shown that it greatly improves the estimated odometry on a real robot. Moreover we shown that the online variant is able to quickly estimate changes of the motion model parameters.

The abstraction of the IMU pre-integration theory has allowed us to apply it to a simpler case, 2D odometry, obtaining easily self-calibration. This abstraction allows us in future work to apply it to different motion models and other self-calibration problems, including in 3D. Furthermore, we plan on integrating the pre-integration scheme presented in this paper to a complete SLAM algorithm.

Appendix I
Jacobians

All Jacobian blocks are 3×3 unless otherwise stated. See [11] for general formulae of Jacobians in $SE(2)$. The Jacobians of the motion arc $\mathbf{b}_k$ in (6) and (7) are,

$$\mathbf{J}^{\mathbf{b}_k}_{\delta\boldsymbol{\psi}_k} = \begin{bmatrix} \frac{c_l r_l}{2} & \frac{c_r r_r}{2} \\ 0 & 0 \\ -\frac{c_l r_l}{c_d d} & \frac{c_r r_r}{c_d d} \end{bmatrix} \in \mathbb{R}^{3\times 2} , \tag{27a}$$

$$\mathbf{J}^{\mathbf{b}_k}_{\mathbf{c}} = \begin{bmatrix} \frac{\delta\psi_{l,k} r_l}{2} & \frac{\delta\psi_{r,k} r_r}{2} & 0 \\ 0 & 0 & 0 \\ -\frac{\delta\psi_{l,k} r_l}{c_d d} & \frac{\delta\psi_{r,k} r_r}{c_d d} & -\frac{\delta\theta_k}{c_d} \end{bmatrix} . \tag{27b}$$

The Jacobian of the current delta δ_k (8) is computed from the Jacobian of $\mathrm{Exp}(\mathbf{b})$, which for $\mathbf{b} = (u, v, \theta)^\top \in se(2)$ is,

$$\mathbf{J}^{\mathrm{Exp}(\mathbf{b})}_{\mathbf{b}} = \begin{bmatrix} \frac{\sin\theta}{\theta} & \frac{\cos\theta - 1}{\theta} & \frac{\theta u + v - v\cos\theta - u\sin\theta}{\theta^2} \\ \frac{1-\cos\theta}{\theta} & \frac{\sin\theta}{\theta} & \frac{-u + \theta v + u\cos\theta - v\sin\theta}{\theta^2} \\ 0 & 0 & 1 \end{bmatrix} . \tag{28}$$

The Jacobians $\mathbf{J}^{\Delta_{ik}}_{\Delta_{ij}}, \mathbf{J}^{\Delta_{ik}}_{\delta_k}$ of (9) and $\mathbf{J}^{\mathbf{U}}_{\mathbf{T}}, \mathbf{J}^{\mathbf{U}}_{\widetilde{\Delta}^S}, \mathbf{J}^{\mathbf{L}}_{\Delta}, \mathbf{J}^{\mathbf{L}}_{\mathbf{T}}$ of (16) are computed from those of the $SE(2)$ composition $\mathbf{T}_a \boxplus \mathbf{T}_b$,

$$\mathbf{J}^{\mathbf{T}_a \boxplus \mathbf{T}_b}_{\mathbf{T}_a} = \begin{bmatrix} \mathbf{R}_b^\top & \mathbf{R}_b^\top [1]_\times \mathbf{p}_b \\ \mathbf{0} & 1 \end{bmatrix} , \quad \mathbf{J}^{\mathbf{T}_a \boxplus \mathbf{T}_b}_{\mathbf{T}_b} = \mathbf{I} , \tag{29}$$

for $\mathbf{T}_i = (\mathbf{p}_i, \theta_i)$ and $\mathbf{R}_i = \mathbf{R}(\theta_i)$. Finally in (17) we have

$$\mathbf{J}^{\mathbf{e}}_{\mathbf{U}} = \mathbf{I} , \ \mathbf{J}^{\mathbf{e}}_{\mathbf{L}} = -\mathbf{I} . \tag{30}$$

References

[1] J. Borenstein, "UMBmark: a benchmark test for measuring odometry errors in mobile robots," *Proc. SPIE*, vol. 2591, pp. 113–124, 1995.

[2] A. Kelly, "Fast and easy systematic and stochastic odometry calibration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 4, Sendai, Sep. 2004, pp. 3188–3194.

[3] A. Martinelli and R. Siegwart, "Estimating the odometry error of a mobile robot during navigation," in *Proc. 1st Eur. Conf. Mobile Robots*, Radziejowice, Sep. 2003.

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

[5] A. Censi, A. Franchi, L. Marchionni, and G. Oriolo, "Simultaneous Calibration of Odometry and Sensor Parameters for Mobile Robots," *IEEE Trans. Robotics*, vol. 29, no. 2, pp. 475–492, apr 2013.

[6] R. Kummerle, G. Grisetti, C. Stachniss, and W. Burgard, "Simultaneous parameter calibration, localization, and mapping for robust service robotics," in *Proc. IEEE Workshop Adv. Robotics Soc. Impacts*, Half-Moon Bay, CA, Oct. 2011, pp. 76–79.

[7] M. D. Cicco, B. D. Corte, and G. Grisetti, "Unsupervised calibration of wheeled mobile platforms," in *Proc. IEEE Int. Conf. Robotics Autom.*, Stockholm, May 2016, pp. 4328–4334.

[8] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 61–76, 2012.

[9] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[10] D. Atchuthan, A. Santamaria-Navarro, N. Mansard, O. Stasse, and J. Solà, "Odometry based on auto-calibrating inertial measurement unit attached to the feet," in *2018 European Control Conference (ECC)*, June 2018, pp. 3031–3037.

[11] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," Institut de Robòtica i Informàtica Industrial, Barcelona, Tech. Rep. IRI-TR-18-01, 2018.

[12] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[13] M. Jaimez, J. G. Monroy, and J. González-Jiménez, "Planar odometry from a radial laser scanner. A range flow-based approach," in *Proc. IEEE Int. Conf. Robotics Autom.*, Stockholm, May 2016, pp. 4479–4485.

[14] M. Quigley, B. Gerkey, K. Conley, T. F. J. Faust, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *Proc. IEEE ICRA Workshop Open Source Soft. Robot.*, Kobe, 2009.