Simultaneous Parameter Calibration, Localization, and Mapping

Rainer Kümmerle[1]   Giorgio Grisetti[2,1]   Wolfram Burgard[1]

[1] *University of Freiburg, Georges-Koehler-Allee 079, D-79110 Freiburg,*

[2] *La Sapienza University of Rome, Via Ariosto 25, I-00185 Rome*

{kuemmerl,grisetti,burgard}@informatik.uni-freiburg.de

Abstract

The calibration parameters of a mobile robot play a substantial role in navigation tasks. Often these parameters are subject to variations that depend either on changes in the environment or on the load of the robot. In this paper, we propose an approach to simultaneously estimate a map of the environment, the position of the on-board sensors of the robot, and its kinematic parameters. Our method requires no prior knowledge about the environment and relies only on a rough initial guess of the parameters of the platform. The proposed approach estimates the parameters on-line and it is able to adapt to non-stationary changes of the configuration. We tested our approach in simulated environments and on a wide range of real world data using different types of robotic platforms.

keywords: calibration, SLAM, mapping

1   Introduction

Many approaches to navigation tasks such as localization, path planning, motion control, and simultaneous localization and mapping (SLAM) rely on the knowledge of specific parameters of the robot. These parameters typically include the position of the sensor on the platform or specific aspects of the kinematic model that translates encoder ticks into a relative movement of the mobile base.

The influence of the parameters on the accuracy of state estimation processes can be substantial. For instance, an accurate calibration of the odometry can seriously improve the expected accuracy of the motion prediction by reducing the search space of the algorithms that provide the motion estimates. Figure 1 shows a motivating example. Here, we ran a scan-matching algorithm given the odometry measurements of a robot that moves along a corridor. Since the corridor is not rich in features, the scan matcher yields solutions that are highly ambiguous along the direction of the corridor. As a result, scan-matching approaches tend to make corridors "shorter" [1]. To limit this effect one can restrict the
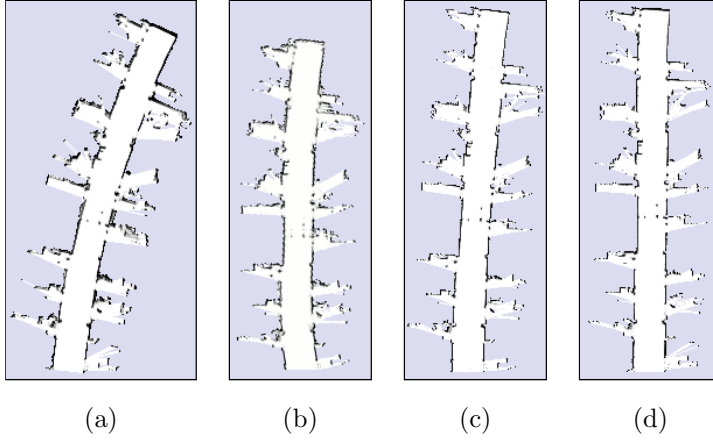

(a) (b) (c) (d)

Figure 1: (a) Map obtained based on the raw uncalibrated odometry of a robot with unevenly inflated tires traveling along a corridor. The result of applying a scan-matching algorithm with a large search space to account for the uncalibrated odometry leads to the shortened map shown in (b). A restriction of the search space is not able to fully correct the errors as visualized in (c). However, applying the correct calibration together with a small search space leads to an accurate estimate as depicted in (d).

search space of the scan-matcher to a small region around the position predicted by odometry. This reduces the computational requirements but requires a highly accurate calibration of the odometry.

While 2D range scans do not provide highly distinguishable features, camera images allow to extract discriminable features, e.g., SIFT [2] or SURF [3]. However, the best match of a feature is not necessarily the correct match. This leads to a set of feature matches which contains a substantial amount of outliers. Typically, an algorithm based on Random Sample Consensus (RANSAC) [4] is employed to robustly estimate the inlier set despite the large amount of outliers. If an initial guess of the camera motion is available, for instance, by the odometry of the robot, we are able to restrict the search for the match to the most likely area in the image. Figure 2 visualizes the number of correct feature matches returned by the matching algorithm with and without access to prior information on the motion of the camera. As we can see, the number of correct matches using the prior information is substantially larger compared to an uninformed matching algorithm. A larger number of correct matches in principle yields a better estimate for the transformation between the two frames. To accurately predict the motion of the camera based on the odometry, the robot requires to know the parameters of the odometry and the position of the sensor.

To obtain these parameters it is common to either rely on the specifications of the platform, to manually measure them, or to run ad-hoc calibration procedures before the mission of the robot is started. The latter solution is typically the most accurate and robust, since it can exploit the a priori knowledge of a calibration pattern to infer reasonable initial guesses for the parameters. However, ad-hoc calibration procedures suffer from two main drawbacks: they are not able to estimate non-stationary parameters and they need to be repeated whenever there is a potential change in the robot configuration. On the one hand, these changes will happen unavoidably during the lifetime of the robot
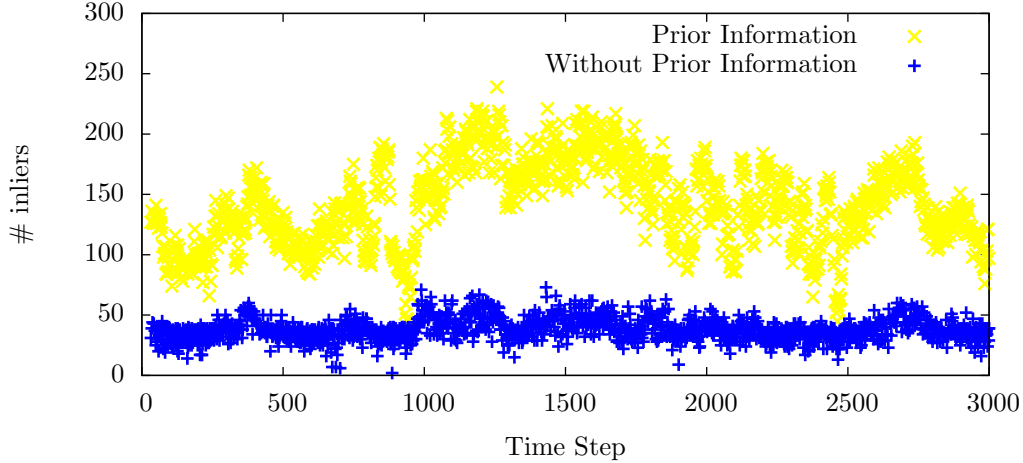
Figure 2: The number of inliers of a matching algorithm operating on feature matches. A matching algorithm that considers the prior information about the motion of the camera achieves a larger set of inliers compared to an algorithm which has no access to this information.

as a consequence of the wearing of mechanical parts. On the other hand, it is not infrequent to observe sudden changes in these parameters as a consequence of particular events. As an example, the odometry parameters depend both on the distribution of the load on the platform and on the type of surface the robot moves upon. These external quantities affect both the effective wheel radii and the accuracy of the odometry prediction. When the robot carries a load the odometry will change, and similarly when it moves from carpet to concrete. One solution to dynamically estimate these parameters is to treat them as hidden state variables that have to be estimated together with the map and the position of the robot. To this end, we could use special equipment such as an external position tracking device. More promising, however are calibration procedures that only rely on the data gathered by the robot and do not require any preparation or additional information.

In this paper, we present an approach to estimate the calibration parameters of a robot equipped with an on-board sensor and wheel encoders while it performs SLAM. The core idea of our method is to treat the map estimate as a calibration pattern and to constantly refine the estimates of the map, the trajectory and parameters of the robot through a least-squares procedure. If the map and the robot positions are known, our method behaves as a standard least-squares approach for parameter calibration. We model the problem as a hyper-graph, where each node represents either a robot position, the sensor position on the robot, or the kinematic parameters of the odometry. Our approach allows to determine these state variables on the fly (e.g., sensor positions and odometry calibration). To deal with temporal changes or more in general with interdependencies between the parameters and the other state variables, we estimate the parameters on the most recent data. This approach allows a mobile robot, for example, to estimate a different set of odometry parameters for different regions of the environment and to better model the motion of the robot in these areas. Our approach might additionally be beneficial in a variety of contexts including, for instance, terrain classification. We present evaluations of our approach in

simulated and a wide range of real world experiments using several robot platforms moving on different types of ground.

2 Related Work

The traditional approaches to calibrate a mobile robot and its sensors involve to accurately measure the trajectory of the robot while recording odometry and sensor measurements, for example, by external cameras or lasers [5]. To calibrate the sensor position, they match the measurements against a known map to recover the trajectory of the sensor and use a least squares estimator to determine the relative transformation between the robot and its sensor. In a similar way the odometry parameters can be estimated via another independent least-squares estimator given the knowledge of the reference trajectory [6]. For example, Antonelli *et al.* [7, 8] considered an external camera to track the position of a robot for calibrating the odometry parameters of a differential drive. Their method employs a least squares estimator which exploits the linear relation between the measurements and the unknown parameters.

In the context of computer vision, the idea to calibrate the intrinsic camera parameters while performing structure from motion is commonly known [9]. This problem has a structure which is very similar to the one addressed by our method. The main difference lies in the kind of parameters that are estimated. One of the first approaches to determine the stationary parameters of a mobile robot and to determine the error of the motion was proposed by Martinelli and Siegwart [10]. The idea behind this work is to extend the state of a Kalman filter used for localization with the kinematic parameters of the odometry. Whereas this approach can operate on-line, it requires an a-priori known map. Subsequently, Jones *et al.* [11] and Kelly and Sukhatme [12] extended the EKF and UKF algorithm to include calibration parameters. Despite their increased complexity, in these non-linear problems smoothing approaches outperform filtering methods in terms of accuracy.

Eliazar and Parr [13] proposed to use an EM approach to learn the motion model for a mobile robot. Their method is able to accurately estimate the parameters and it does not require to know the map in advance. However, their approach requires to be run off-line due to its high computational requirements. Based on a localization algorithm Roy and Thrun [14] estimate online the systematic error in the odometry. They treat the error in translation and rotation independently. Both approaches model the calibration as a linear function of the odometry measurement, whereas our approach estimates the physical parameters of the robot.

Gao and Spletzer [15] presented an approach to determine the extrinsic calibration parameters between two laser range finders. Underwood *et al.* [16] proposed a method to determine the 3D position of a laser within the body frame of the robot. Compared to our method those approaches either rely on establishing feature correspondences between the individual observations by preparing the environment with laser-reflective tape or assume a simple and partially known geometric environment for calibrating the sensors. Assuming a known configuration of landmarks in the environment, Antonelli *et al.* [17]

described an approach to calibrate the odometry together with the extrinsic and intrinsic parameters of a camera mounted on the robot.

Censi *et al.* [18] proposed a technique similar to our method. They construct a least squares calibration problem that estimates both the kinematic parameters and the sensor position. Their approach does not need to know the map in advance, but it is restricted to the estimation of stationary parameters. Furthermore, since it relies on scan-matching to estimate the ego-motion of the sensor, this method does not provide an accurate map in large and loopy environments. However, such a pure calibration method or one of the techniques described above are the best choice, if no simultaneous mapping of the environment is required. Here, calibration methods may provide guarantees for the convergence or the optimality of the parameters. Thus, these techniques do not require suitable initial values for the parameters.

Our work can be seen as an extension of traditional graph-based SLAM algorithms. These methods model the SLAM problem as a graph, whose nodes represent robot poses and whose edges connect two nodes if there is a measurement involving both of them. Each edge is labeled with the relative transformation between the robot poses. An edge can arise either from matching a pair of observations or from an odometry measurement between consecutive poses. A solution to the problem is a configuration of the nodes that better satisfies the measurements encoded in the constraints. One seminal work in this context is the work of Lu and Milios [19] where the relative motion between two scans is measured by scan-matching and the resulting graph is optimized by iterative linearization. In the past, several methods have been proposed to either optimize the graph on-line and in a faster way [20, 21, 22, 23] or to extract the graph from the raw measurements in more robust ways [24, 25]. All these approaches assume a known calibration of the system.

Note that when we augment the problem with the calibration variables it cannot be described anymore by a graph but instead requires a hyper-graph. This is because a measurement does not only depend on a pair of variables (the connected nodes), but rather on a triplet (the nodes and the calibration parameters). In this paper, we therefore extend the standard graph optimization framework to handle this class of problems. Our approach is able to simultaneously estimate the map of the environment and calibrate the parameters of a robot in a continuous manner. We do not require any special preparation for the environment, such as an external tracking system or landmarks to be placed in the designated area.

3 Simultaneous Calibration, Localization, and Mapping

Our system relies on the graph-based formulation of the SLAM problem to estimate the maximum-likelihood configuration. In contrast to the traditional SLAM methods we explicitly model that the measurements obtained by the robot are given in different coordinate frames. For example, the odometry of the robot is given by the velocity measurements of its wheels. Applying the forward kinematics of the platform allows to transform the velocities measured during a time interval into a relative displacement

of the platform expressed in the odometry frame. Usually, the robot is equipped with a sensor that is able to observe the environment, e.g., a laser range finder. This sensor is mounted on the robot and obtains measurements in its own coordinate frame. Thus, a scan-matching algorithm which aligns two range scans in a common coordinate frame has to project the computed motion estimate through the kinematic chain of the robot to obtain an estimate for the motion of the robot's base. As it is not always easy to measure the offset transformation between the base of the robot and the sensor or to determine the parameters for the forward kinematics, we suggest to integrate those into the maximum likelihood estimation process. In particular, the parameters of the forward kinematics are affected by the wear of the devices during the lifetime of the robot.

3.1   Description of the Hyper-Graph

Whenever the robot obtains a measurement we add a node to the graph. This node represents the position of the robot at which the measurement was obtained. Let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top$ be a vector of parameters, where $\mathbf{x}_i = (x_i, y_i, \theta_i)^\top$ describes the position of node i. Furthermore, let $\mathbf{l}$ be the pose of the on-board sensor relative to the coordinate frame of the robot and let $\mathbf{z}_{ij}$ and $\mathbf{\Omega}_{ij}^{\mathbf{z}}$ be respectively the mean and information matrix of an observation of node j seen from node i. Finally, let $\mathbf{k}$ be the parameters of the forward kinematics function and $\mathbf{u}_i$ and $\mathbf{\Omega}_i^{\mathbf{u}}$ be respectively the motion command and the information matrix which translates the robot from node i to $i+1$.

The error function $\mathbf{e}_i^{\mathbf{u}}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{k}, \mathbf{u}_i)$ measures how well the parameter blocks $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{k}$ satisfy the odometry measurement $\mathbf{u}_i$. A value of $\mathbf{0}$ means that the constraint is perfectly satisfied by the parameters. For simplicity of notation, we will encode the involved quantities in the indices of the error function:

$$\mathbf{e}^{\mathbf{u}}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{k}, \mathbf{u}_i) \overset{\text{def.}}{=} \mathbf{e}^{\mathbf{u}}(\mathbf{x}_i, \mathbf{x}_{i+1}) \overset{\text{def.}}{=} \mathbf{e}_i^{\mathbf{u}}(\mathbf{x}). \tag{1}$$

The error function $\mathbf{e}_i^{\mathbf{u}}(\mathbf{x})$ is defined as

$$\mathbf{e}_i^{\mathbf{u}}(\mathbf{x}) = (\mathbf{x}_{i+1} \ominus \mathbf{x}_i) \ominus K(\mathbf{u}_i, \mathbf{k}), \tag{2}$$

where $K(\cdot)$ is the forward kinematics function converting from wheel velocities to a relative displacement of the vehicle. Furthermore, $\oplus$ is the usual motion composition operator [26] and $\ominus$ its inverse:

$$\mathbf{x}_{i+1} \ominus \mathbf{x}_i \overset{\text{def.}}{=} (\mathbf{x}_i)^{-1} \oplus \mathbf{x}_{i+1}. \tag{3}$$

For a robot with a differential drive, which is one of the most common types of robots, the odometry $\mathbf{u} = (v_l, v_r)^\top$ consists of the velocities of the left and the right wheel. The wheel velocities are computed by counting the encoder ticks of the motors during the time step which are multiplied by the respective radii r_l and r_r of the wheels. Furthermore, the distance b between the two wheels has to be known to compute the circular arc on which the robot moves. The relative motion during the time interval Δt is
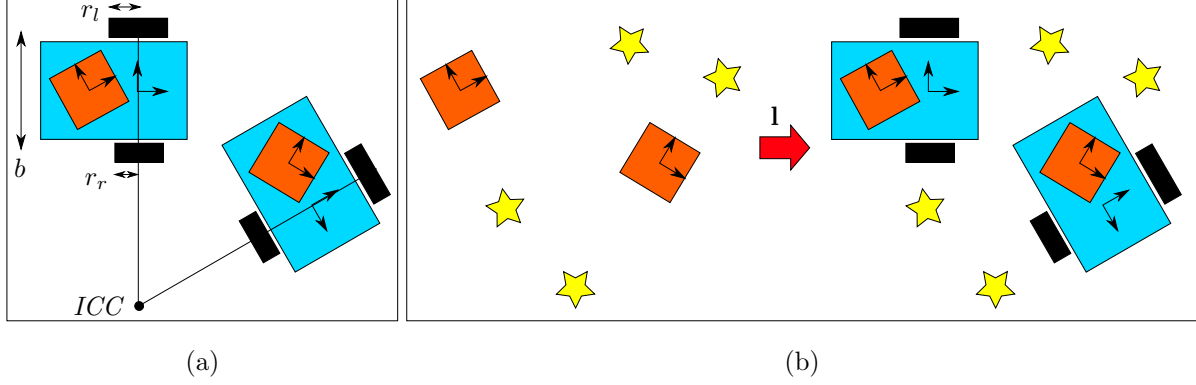
Figure 3: (a) The parameters $(r_r, r_l, b)^\top$ used to compute the motion of the robot given the wheel velocities. (b) The sensor observations (indicated as yellow stars) allow to estimate the ego-motion of the sensor and the motion of the robot *given* the sensor offset **l**.

given by

$$K(\mathbf{u}, \mathbf{k}) = \begin{pmatrix} R(\Delta t \omega) & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} -ICC \\ 0 \end{pmatrix} + \begin{pmatrix} ICC \\ \Delta t \omega \end{pmatrix}, \tag{4}$$

where $R(\cdot)$ is the 2D rotation matrix of its argument, $ICC = (0, \frac{b}{2} \frac{r_l v_l + r_r v_r}{r_l v_l - r_r v_r})^\top$, and $\omega = \frac{r_l v_l - r_r v_r}{b}$. Thus, the calibration parameter $\mathbf{k} = (r_r, r_l, b)^\top$ for the odometry is a three-dimensional vector (see Figure 3a for an illustration).

Additionally, the error function $\mathbf{e}^\mathbf{l}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{l}, \mathbf{z}_{ij})$ measures how well the parameter blocks $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{l}$ satisfy the constraint $\mathbf{z}_{ij}$. If the three parameters perfectly satisfy the error function, then its value is $\mathbf{0}$. Here, we assume that the laser is mounted without inclination which is the ideal condition (see Figure 3b). The error function $\mathbf{e}^\mathbf{l}_{ij}(\mathbf{x})$ has the following form:

$$\mathbf{e}^\mathbf{l}_{ij}(\mathbf{x}) = ((\mathbf{x}_j \oplus \mathbf{l}) \ominus (\mathbf{x}_i \oplus \mathbf{l})) \ominus \mathbf{z}_{ij}. \tag{5}$$

In Eq. (5) we applied the same simplifying notation as defined in Eq. (1).

The goal of our maximum likelihood approach is to find the configuration of $[\mathbf{x}^*, \mathbf{l}^*, \mathbf{k}^*]$ which minimizes the negative log-likelihood $\mathbf{F}(\mathbf{x}, \mathbf{l}, \mathbf{k})$ given all the observations

$$\mathbf{F}(\mathbf{x}, \mathbf{l}, \mathbf{k}) = \sum_{\langle i,j \rangle} \mathbf{e}^\mathbf{l}_{ij}(\mathbf{x})^\top \mathbf{\Omega}^\mathbf{z}_{ij} \mathbf{e}^\mathbf{l}_{ij}(\mathbf{x}) + \sum_i \mathbf{e}^\mathbf{u}_i(\mathbf{x})^\top \tilde{\mathbf{\Omega}}^\mathbf{u}_i \mathbf{e}^\mathbf{u}_i(\mathbf{x}), \tag{6}$$

where $\tilde{\mathbf{\Omega}}^\mathbf{u}_i$ is the projection of $\mathbf{\Omega}^\mathbf{u}_i$ through the forward kinematics function $K(\cdot)$ via the unscented transformation [27]. Since the projection depends on the estimate of $\mathbf{k}$, we update the projection if $\mathbf{k}$ changes substantially.

Given this formulation we may easily integrate prior knowledge, for example, the manually — thus non-precisely — measured transformation of the laser. This is possible as long as the prior information can be represented by a Gaussian distribution. Furthermore, state transitions observed by measurements, e.g., the robot actively rotates the laser scanner, can be incorporated.

To estimate the calibration parameters, the trajectory of the robot should introduce measurements that constrain all possible dimensions of $\mathbf{k}$ and $\mathbf{l}$. Clearly, a trajectory only consisting of straight line motions does not allow to observe the position of the laser. The same holds for a perfectly circular trajectory, since the laser could be anywhere on the circle. Both cases are pathologic and can easily be avoided by varying the wheel velocities of the robot. Recently, Censi *et al.* [28] gave a proof that our informal statement holds. They show that two linear independent velocity commands which lead to two circular arcs witch different radii are sufficient to guarantee the observability of the parameters.

3.2   3D on-board sensors

In the previous section we described how to estimate the position of an on-board 2D range scanner. However, with the advent of the Microsoft Kinect an inexpensive alternative exists which provides dense 3D range data along with an RGB image. Such depth cameras emit an infrared light pattern which is received by a camera. Computing the disparity of features in the images allows – similar to a stereo camera pair – to obtain 3D depth measurements.

As with 2D range scanners, we are able to estimate the 3D ego-motion $\mathbf{z}_{ij}$ of the camera between two frames by aligning the measurements. Along with the 3D offset of the sensor $\mathbf{c}$ in the coordinate frame of the robot we are able to constrain the motion of the robot:

$$\mathbf{e}_{ij}^{\mathbf{c}}(\mathbf{x}) = ((\text{toSE3}(\mathbf{x}_j) \oplus \mathbf{c}) \ominus (\text{toSE3}(\mathbf{x}_i) \oplus \mathbf{c})) \ominus \mathbf{z}_{ij}, \tag{7}$$

where $\text{toSE3}(\cdot)$ projects its argument from 2D to 3D, i.e., the pose is extended with a z coordinate and pitch/roll angles which are all set to 0.

Eq. (7) does not observe the height of the sensor above the ground. Typically a rough initial guess for the attitude of the sensor is easy to obtain manually. Assuming such an estimate is available, we are able to extract the ground plane observed by the sensor. To this end, a RANSAC algorithm samples three points from the point cloud to calculate a candidate plane. If the angle of the normal vector of the candidate lies within a threshold (20 degrees in our current implementation) around the ground plane assumption given by the prior, we determine the points whose distance to the plane is smaller than 0.1 m. The candidate with the largest inlier set yields our estimate of the ground plane. The distance of the sensor to the extracted ground plane results in our observation for the height of the sensor. Additionally, the ground plane $\mathbf{p}$ allows us to further constrain the attitude of the sensor, namely the rotation of the sensor with respect to the ground plane. Hence, we obtain the additional error term

$$\mathbf{e}_i^{\mathbf{p}}(\mathbf{c}) = \mathbf{c} \ominus \mathbf{p}_i, \tag{8}$$

which constrains the pitch/roll angles of the sensor to match the ground plane observation.

Replacing the sensor error term and adding Eq. (8) to Eq. (6), we obtain the negative log-likelihood $\mathbf{F}(\mathbf{x}, \mathbf{c}, \mathbf{k})$ given all the observations

$$\mathbf{F}(\mathbf{x}, \mathbf{c}, \mathbf{k}) \quad = \quad \sum_{\langle i,j \rangle} \mathbf{e}_{ij}^{\mathbf{c}}(\mathbf{x})^{\top} \mathbf{\Omega}_{ij}^{\mathbf{z}} \mathbf{e}_{ij}^{\mathbf{c}}(\mathbf{x}) + \sum_{i} \mathbf{e}_i^{\mathbf{u}}(\mathbf{x})^{\top} \tilde{\mathbf{\Omega}}_i^{\mathbf{u}} \mathbf{e}_i^{\mathbf{u}}(\mathbf{x}) + \sum_{i} \mathbf{e}_i^{\mathbf{p}}(\mathbf{c})^{\top} \mathbf{\Omega}_i^{\mathbf{p}} \mathbf{e}_i^{\mathbf{p}}(\mathbf{c}), \tag{9}$$

where $\mathbf{\Omega}_i^{\mathbf{P}}$ represents the information matrix of the ground plane observation. Minimizing Eq. (9) yields the poses of the robot, the calibration parameters of the forward kinematics function, and the 3D offset of the range sensor.

3.3 Estimation via Least Squares on a Hyper Graph

Without loss of generality we will refer to the whole state vector which is either $[\mathbf{x} \ \mathbf{l} \ \mathbf{k}]$ or $[\mathbf{x} \ \mathbf{c} \ \mathbf{k}]$ as $\mathbf{y}$, without distinguishing the parameter blocks. Additionally, we identify each hyper-edge by a unique index k instead of the pair of indices i, j and the superscript letter as in Eq. (6) and Eq. (9).

We initialize the optimization problem as follows. The poses of the robot represented by $\mathbf{x}$ are initialized by the odometry measurements whereas the specifications of the robot are considered as initial values of $\mathbf{k}$. For the position $\mathbf{l}$ of the sensor a rough initial guess is required which is easy to obtain manually. Given such an initial guess $\check{\mathbf{y}}$ of the parameters is known, a numerical solution of Eq. (6) can be obtained by using the popular Gauss-Newton or Levenberg-Marquardt (LM) algorithms [29, §15.5]. The idea is to approximate the error function by its first order Taylor expansion around the current initial guess $\check{\mathbf{y}}$

$$
\begin{aligned}
\mathbf{e}_k(\check{\mathbf{y}}_k \boxplus \mathbf{\Delta y}_k) &= \mathbf{e}_k(\check{\mathbf{y}} \boxplus \mathbf{\Delta y}) & (10) \\
&\approx \mathbf{e}_k + \mathbf{J}_k \mathbf{\Delta y}. & (11)
\end{aligned}
$$

Here, $\mathbf{J}_k$ is the Jacobian of $\mathbf{e}_k(\mathbf{y} \boxplus \mathbf{\Delta y})$ with respect to $\mathbf{\Delta y}$ computed in $\mathbf{\Delta y} = 0$ and $\mathbf{e}_k \overset{\text{def.}}{=} \mathbf{e}_k(\check{\mathbf{y}})$. Furthermore, $\boxplus$ is an operator that applies the increments $\mathbf{\Delta y}$ to the current state $\check{\mathbf{y}}$. This accounts for over-parameterized states, and can better deal with non Euclidean state spaces. Clearly, if both $\check{\mathbf{y}}$ and $\mathbf{\Delta y}$ are Euclidean the $\boxplus$ degenerates to a regular $+$. Substituting Eq. (11) in the error terms $\mathbf{F}_k$ of Eq. (6), we obtain the following quadratic form

$$
\mathbf{F}(\mathbf{y}) = \mathbf{c} + 2\mathbf{b}^T \mathbf{\Delta y} + \mathbf{\Delta y}^T \mathbf{H} \mathbf{\Delta y} \tag{12}
$$

that can be minimized in $\mathbf{\Delta y}$ by solving the system

$$
(\mathbf{H} + \lambda \mathbf{I}) \mathbf{\Delta y}^* = -\mathbf{b}. \tag{13}
$$

Here, $\mathbf{H}$ and $\mathbf{b}$ are respectively the approximated Hessian of the error function and the error gradient:

$$
\begin{aligned}
\mathbf{H} &= \sum_k \mathbf{J}_k^T \mathbf{\Omega}_k \mathbf{J}_k & (14) \\
\mathbf{b} &= \sum_k \mathbf{J}_k^T \mathbf{\Omega}_k \mathbf{e}_k. & (15)
\end{aligned}
$$

The quantity λ is a damping factor: the larger λ is the smaller are the $\mathbf{\Delta y}$. This is useful to control the step size in case of non-linear surfaces. The idea behind the LM algorithm is to dynamically control the damping factor. At each iteration the error of the new configuration is monitored. If the new error is lower than the previous one, λ is decreased for the next iteration. Otherwise, the solution is reverted and λ is increased. For $\lambda = 0$ we obtain the Gauss-Newton method.

Whenever the increments $\boldsymbol{\Delta}\mathbf{y}$ are computed by solving Eq. (13), they can be applied to the previous solutions by means of the $\boxplus$ operator

$$\breve{\mathbf{y}} \quad \leftarrow \quad \breve{\mathbf{y}} \boxplus \boldsymbol{\Delta}\mathbf{y}^*. \tag{16}$$

The $\boxplus$ operator is able to handle singularities and over-parameterized state variables. We refer the reader to [30] for a more detailed description. In our case we employ the motion composition operator $\oplus$ to apply the update for the parameters blocks $\mathbf{x}$, $\mathbf{l}$ and $\mathbf{c}$, whereas $\mathbf{k}$ is updated by a regular $+$. The LM algorithm iterates the linearization in Eq. (11), the construction of the linear system and its solution (Eq. (13)), and the update step until some termination criterion is reached.

From Eq. (6) we notice that each of the terms in the sum depends on at most three parameter blocks. More precisely, if the constraint k arises from an odometry measurement, it will depend on the connected robot poses $\mathbf{x}_i$ and $\mathbf{x}_j$ and by the odometry parameters $\mathbf{k}$. Alternatively, if a constraint arises from a laser measurement, it will depend on the connected robot poses $\mathbf{x}_i$ and $\mathbf{x}_j$ and on the laser position $\mathbf{l}$. Accordingly, each of the Jacobians in Eq. (14) and in Eq. (15) will have only three non-zero blocks, in correspondence of the variables involved by the constraint. Thus, each of the terms in the sum of Eq. (14) will be a matrix with at most nine non-zero components. Furthermore, $\mathbf{H}$ will have a number of non-zero entries proportional to the number of constraints. This results in a sparse system that can be efficiently solved. To optimize it, we employ the g^2o toolkit [31] which allows us to solve one iteration of a calibration problem having 3,000 nodes in less than 0.01 s using one core of an Intel i7@2.8 GHz.

3.4 Monitoring the Convergence

Some calibration parameters may be constant while others change. For example, the laser position $\mathbf{l}$ is constant if the robot has no actuator to move this sensor. Therefore, it is of interest to decide whether enough data has been collected so that one can stop calibrating and reduce the computational demands. To this end, we can consider the approximated Hessian $\mathbf{H}$ and compute the marginal covariance of the calibration parameters. The marginal covariance $\boldsymbol{\Sigma}_\mathbf{l}$ of the calibration parameter is given by extracting the corresponding block of $\mathbf{H}^{-1}$. The matrix $\mathbf{H}$ is sparse, symmetric, and positive definite, thus Cholesky decomposition can be applied to factorize $\mathbf{H}$. By applying an algorithm based on dynamic programming (see [32]) we can efficiently compute the desired elements of $\mathbf{H}^{-1}$ given the Cholesky factor. As we will show in the experiments, this information allows us to access the quality of the estimated parameters.

4 Experiments

The approach described above has been implemented and evaluated on both simulated and real-world data acquired with a heterogeneous set of robots equipped with laser range finders. Figure 4 visualizes the robots we used to collect the real-world data considered in this paper.
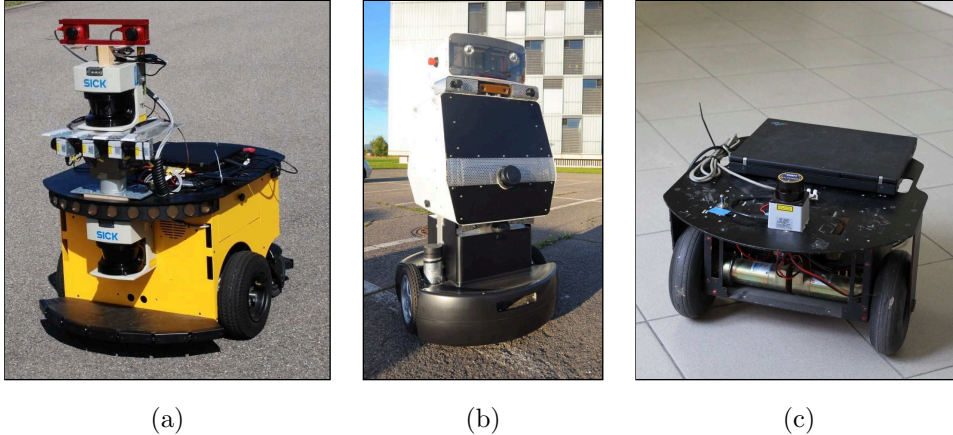
(a)　　　　　　　　　(b)　　　　　　　　　(c)

Figure 4: The robots used to acquire the real-world data sets: (a) MobileRobots PowerBot (b) EUROPA, a custom made platform (c) Pioneer.

The laser based SLAM front-end for processing the data is an own implementation of the framework described by Olson [33] which employs a correlative scan-matcher to estimate the transformation of the laser along with the 3×3 covariance matrix representing the uncertainty of the estimated transformation. The correlative scan-matcher performs an exhaustive search to determine the best fitting alignment for two laser scans within a given search radius. We add a new node to the graph whenever the ego-motion of the laser estimated by the scan-matcher is larger than $0.1\,\mathrm{m}$ in translation or $10°$ in rotation, whichever occurs first.

For estimating the ego-motion of a Kinect depth sensor or a stereo rig, we extract visual features [3] from the images. The transformation between images is estimated by a standard 3-point RANSAC algorithm [34]. Finally, a least-square estimate which minimizes the re-projection error of the feature correspondences is performed along with removing feature projections having large errors from the system for an increased robustness against outliers. Again, a node is inserted into the graph whenever one of the above mentioned thresholds is reached.

4.1   Online odometry calibration

In real world scenarios the odometry is affected by different factors. For example, if the robot is carrying a load, the additional weight compresses inflated tires and results in reduced wheel radii. To this end, we used the PowerBot platform (see Figure 4a) which has a maximum payload of $100\,\mathrm{kg}$ to carry a load of approximately $40\,\mathrm{kg}$. The wheels of the PowerBot are inflated tires whose radii are affected by both the air-pressure of the tires and the total weight of the platform. The load in this set of experiments was intentionally placed on the left hand side of the robot. In a first experiment we recorded datasets in which the robot was either carrying the load or it was operating in its normal configuration. We used one data set for estimating the parameters and a different one for evaluating the odometry calibration parameters. Our approach estimated wheel radii of $r_r = 0.1251\,\mathrm{m}$, $r_l = 0.1226\,\mathrm{m}$ for the normal configuration of the robot and $r_r = 0.1231\,\mathrm{m}$, $r_l = 0.1223\,\mathrm{m}$ while carrying the load. The difference seems to be small,
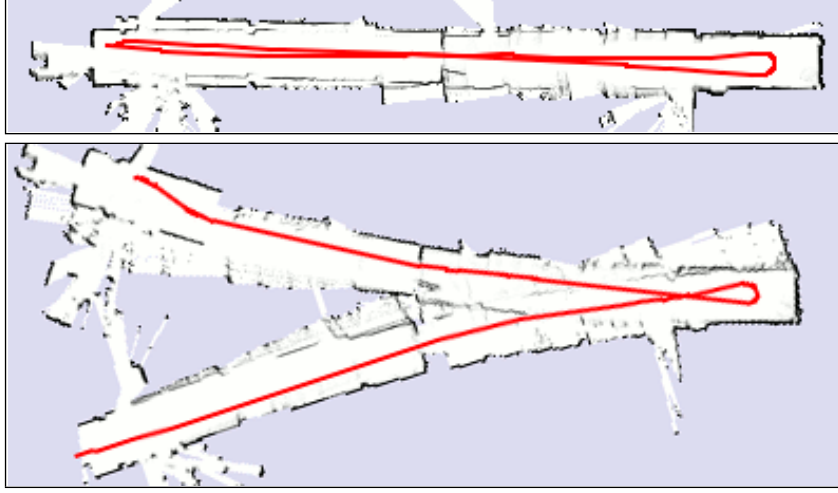
Figure 5: Robot driving up and down a corridor. Top: Applying the calibration corresponding to the current configuration of the robot leads to a good odometry estimate. Bottom: If the robot is carrying a load, the same calibration parameters results in a severe drift in the odometry.
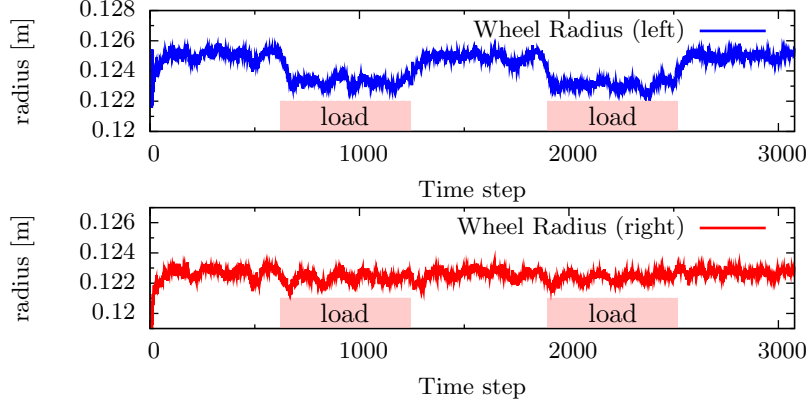

Figure 6: Results of the online estimation of the wheel radii. The robot had to carry a load twice which was placed on the left hand side of the platform leading to a compression of the left wheel.

although it has a substantial effect. Figure 5 shows the outcome of applying the estimate of the normal configuration to the robot carrying the load. Applying the wrong calibration parameter has a crucial effect on the trajectory as it is estimated by the odometry. Since the weight of the load is mutable and can be placed in an arbitrary position on the robot, the best performance can be obtained by calibrating the odometry parameters while the robot is operating.

By considering the 50 most recent measurements within a sliding window around the current node we are able to estimate the wheel radii online also when they are subject to change due to external factors. The size of the sliding window was determined empirically. Too few measurements do not allow to recover the calibration parameters accurately and a large sliding window has an increased latency if the parameters change. For old odometry measurements outside the sliding window we change the error function $\mathbf{e}_i^{\mathbf{u}}(\mathbf{x})$ to employ a fixed value for $\mathbf{k}$, i.e., we only estimate the physical parameters on the
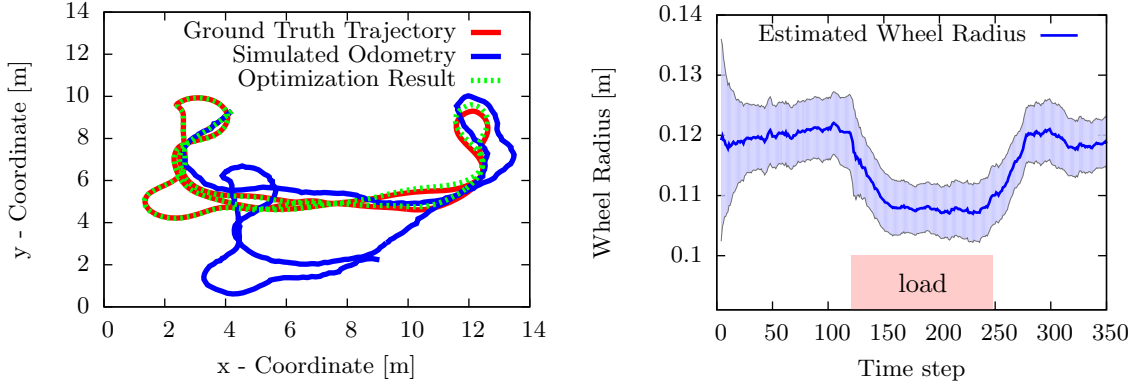
Figure 7: Left: The simulated robot trajectory. Right: Estimating the wheel radii online based on the most recent observations. Here, the robot was carrying a load during the time interval $[120, 240]$ which leads to compressed wheels having smaller radii.

recent data and use the previously estimated parameters to model the odometry error term for older measurements. Figure 6 visualizes the estimated wheel radii during an experiment in which the robot had to carry a load placed on the left hand side of the platform. The robot was carrying the load during the intervals $[600, 1250]$ and $[1865, 2530]$. Using our approach we are able to correctly estimate the wheel radii independent of the load carried by the robot along with the maximum likelihood map of the environment.

As it is hard to obtain ground truth data for real-world data-sets, we simulated a robot traveling on the trajectory depicted in the left part of Figure 7. The environment in which the robot operates consists of two rooms connected by a hallway. Additionally, a laser range finder similar to a SICK LMS is simulated by performing ray-cast operations in the simulation environment. The simulator allows us to directly judge the quality of the calibration results. The odometry measurement and the range measurements obtained by the robot are perturbed by Gaussian noise. Within a simulation experiment we modeled a robot carrying a weight which we simulated having the effect of a reduction of the wheel radius from $0.12\,\mathrm{m}$ to $0.108\,\mathrm{m}$. The robot carries the load during the time interval $[120, 250]$. Figure 7 depicts the results of the online calibration based on the most recent measurements. As we can see, the system is able to represent the compressed wheels and the estimate corresponds well to the ground truth given by the simulator. Furthermore, the trajectory as it is estimated by our approach also matches well to the ground truth as shown in Figure 7.

4.2 Influence of the ground surface

Within real world indoor environments a robot may encounter different floor types, e.g., tiling, PVC flooring, wooden floor, or different types of carpets. To test the influence of the floor type, we recorded data sets in which the robot drives on a soft carpet and on concrete tiling floor, see left image in Figure 8. In this experiment we estimated the odometry parameters online. On both floors the estimated wheel radii were the same. However, by analyzing the standard deviation (see right part of Figure 8) in the
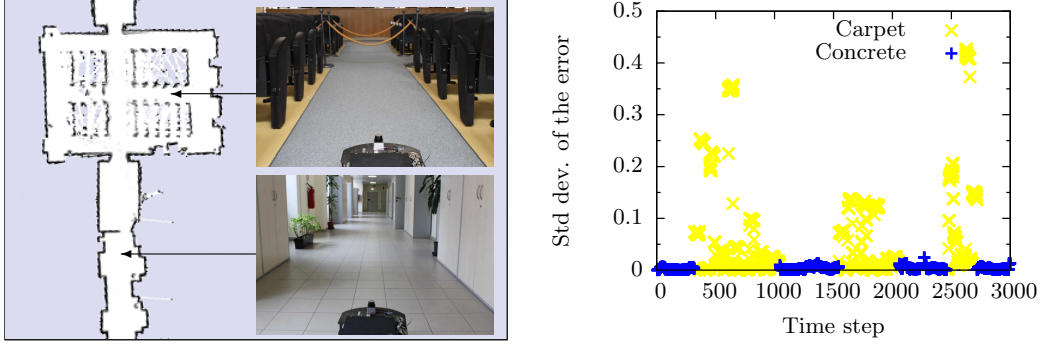
Figure 8: Left: In indoor environments a robot may encounter different floor types. Right: The standard deviation of the error of the odometry edges for the sliding window at each time step.
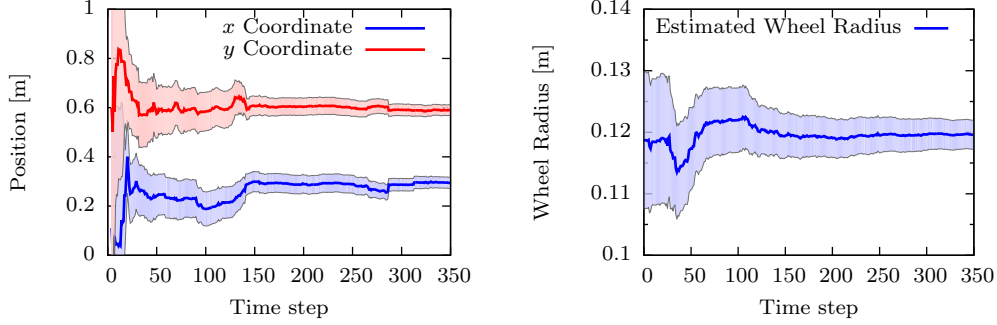

Figure 9: Left: The evolution of the x and y coordinate of the laser transformation as it is estimated by our approach. The true value of the x and y coordinate is $0.3\,\mathrm{m}$ and $0.6\,\mathrm{m}$, respectively. Right: The estimate for the wheel radius having a true value of $0.12\,\mathrm{m}$.

error of the odometry edges $\mathbf{e}_i^{\mathbf{u}}$ for the sliding window around the current node, we observe a higher noise in the odometry due to slippage on the carpet. This information can be stored in the map so that the robot can use it to adjust the motion model noise in a localization task.

4.3   Simulation Experiments

In a first experiment we simulated the 2D laser having a relative transformation of $(0.3, 0.6, 30°)^\top$ with respect to the odometry frame of the robot. Here, we optimized after inserting every node and monitored the evolution of the laser transformation as it is estimated by our approach at each time step. Figure 9 visualizes how the estimate for the x and y coordinate of the relative laser transformation and the radius of the left wheel along with their estimated uncertainty evolves. As we can see the estimate converges quickly to the correct transformation. By monitoring the marginal covariance of the estimated laser transformation we are able to judge the quality of the estimate.

Furthermore, we estimated the odometry parameters of the simulated robot whose left wheel has a radius of $r_l = 0.12\,\mathrm{m}$ whereas the right wheel has a radius of $r_r = 0.125\,\mathrm{m}$. The distance between the wheels is $b = 0.6\,\mathrm{m}$. The output of the calibration is $\hat{r}_l = 0.1207\,\mathrm{m}$, $\hat{r}_r = 0.1264\,\mathrm{m}$, and $\hat{b} = 0.607\,\mathrm{m}$.
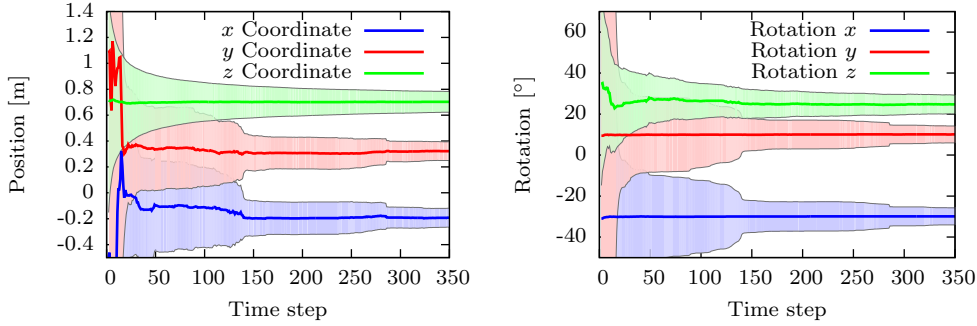
14

Figure 10: The evolution of the position estimate for a simulated depth camera as it is estimated by our approach. The ground-truth values are $(x, y, z) = (-0.2, 0.3, 0.7)$ and $(-30°, 10°, 25°)$ for the rotation around the axes.

Table 1: The parameters of the robots used for our experiments.

	PowerBot	EUROPA	Pioneer
wheel radius [m]	0.125	0.16	0.065
wheel distance [m]	0.56	0.7	0.35
ticks per revolution	22835	20000	1970
laser offset [m, m, °]	(0.22, 0, 0)	(0.3, 0, 0)	(0.1, 0, 0)
laser scanner model	Sick LMS291	Sick LMS151	Hokuyo URG

In a second set of experiments we simulated a robot equipped with a depth camera which estimates its own ego-motion in 3D. The translational offset was simulated as $(-0.2, 0.3, 0.7)^\top$ whereas the rotation around the axes was set to $(-30°, 10°, 25°)^\top$. The simulated wheel radii were the same as in the previous simulation experiment. Additionally, the ground plane observations yielding the rotation of the sensor around the x and y axes and its height above the ground were simulated and perturbed with Gaussian noise, whereas the noise was set to $\mathcal{N}(0, 0.05^2)$ and $\mathcal{N}(0, 1^2)$ for the height estimate and the rotation angles, respectively. Figure 10 visualizes the outcome of our approach. As we can see, the calibration quickly converges to the true values. Furthermore, the estimated values of the kinematics parameters are $\hat{r}_l = 0.1211\,\mathrm{m}$, $\hat{r}_r = 0.1263\,\mathrm{m}$, and $\hat{b} = 0.603\,\mathrm{m}$.

4.4 Real-World Experiments

To evaluate our approach on real-world data we processed data of a heterogeneous set of robots depicted in Figure 4. Table 1 summarizes the parameters of the platforms. To collect the data, we steered each robot twice through the environment. The front-end again processed the data to estimate the motion of the laser for each time step. When the robot re-visits an already known region, the loop closure constraints are added to the graph. Note that the estimation of the calibration parameters does not require to detect loop closures. However, such a constraint allows to reduce the residual error in the trajectory as it is estimated by our approach. Table 2 summarizes the calibration results. As we can see,

Table 2: Calibration results for different robot data sets.

	laser offset (m, m, °)	wheel radii (m, m)	distance m
PowerBot - 1	(0.2258, 0.0026, 0.099)	(0.1263, 0.1275)	0.5825
PowerBot - 2	(0.2231, -0.0031, 0.077)	(0.1243, 0.1248)	0.6091
EUROPA - 1	(0.3067, -0.0051, -0.357)	(0.1603, 0.1605)	0.6969
EUROPA - 2	(0.3023, -0.0087, -0.013)	(0.1584, 0.1575)	0.7109
Pioneer - 1	(0.1045, 0.009, -0.178)	(0.0656, 0.065)	0.3519
Pioneer - 2	(0.1066, -0.0031, -0.28)	(0.0658, 0.0655)	0.3461

Table 3: Calibration results for different robot data sets with a 3D on-board sensor.

	sensor offset (m, m, m, °, °, °)	wheel radii (m, m)	distance m
PowerBot - 1	(0.273, -0.049, 1.005, -0.842, 21.820, 15.673)	(0.1263, 0.1260)	0.5895
PowerBot - 2	(0.284, -0.035, 0.999, -0.645, 21.224, 15.749)	(0.1254, 0.1252)	0.5952
EUROPA - 1	(0.214, 0.061, 1.187, -0.912, 21.402, -0.842)	(0.1559, 0.1557)	0.7039
EUROPA - 2	(0.212, 0.057, 1.185, -0.523, 21.919, -0.612)	(0.1558, 0.1558)	0.7146

the result for the laser transformation are within a few millimeters of the manually measured position. The same holds for the radii of the wheels and their distance to each other.

Additionally, we mounted a Microsoft Kinect on the PowerBot platform and considered the stereo data captured by the EUROPA robot with its Bumblebee stereo camera to evaluate our approach for calibrating the position of those sensors. To this end, we again recorded two data sets with each platform. For the Kinect we manually measured a translation of $(0.28, 0.04, 1.0)$ and a rotation of $(0°, 21°, 15°)$, whereas the translation of the Bumblebee is $(0.21, 0.06, 1.19)$ and the rotation is $(0°, 20°, 0°)$ according to the CAD drawings of the robot. The parameters as they are estimated by our approach are summarized in Table 3. The obtained results indicate that our approach is able to accurately estimate the position of an on-board 3D range sensor and simultaneously calibrate the odometry parameters.

5   Conclusions

In this paper, we presented an approach to estimate the calibration parameters while performing SLAM. Our approach extends the graph-based formulation of the SLAM problem to handle the calibration parameters. The overall approach is accurate and designed for online operation, which allows us to handle changes in the parameters, for example, induced by placing a load onto the robot. Furthermore, compared to ad-hoc calibration methods our approach solely relies on the on-board sensors of the robot and does not require external information.

Additionally, our approach has the potential to provide useful information about the ground surface which affects the uncertainty of the odometry measurements. This information may in the future be exploited for terrain classification and might also be considered by localization algorithms.

Acknowledgments

REFERENCES

[1] C. Stachniss, M. Bennewitz, G. Grisetti, S. Behnke, and W. Burgard. How to learn accurate grid maps with a humanoid. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[2] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal on Computer Vision*, 60(2):91–110, November 2004.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[5] S. Ceriani et al. RAWSEEDS ground truth collection systems for indoor self-localization and mapping. *Journal of Autonomous Robots*, 27(4), 2009.

[6] B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[7] G. Antonelli, S. Chiaverini, and G. Fusco. A calibration method for odometry of mobile robots based on the least-squares technique: theory and experimental validation. *IEEE Trans. on Robotics*, 21(5):994–1004, Oct. 2005.

[8] G. Antonelli and S. Chiaverini. Linear estimation of the odometric parameters for differential-drive mobile robots. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004.

[10] A. Martinelli and R. Siegwart. Estimating the odometry error of a mobile robot during navigation. In *Proc. of the European Conference on Mobile Robots (ECMR)*, 2003.

[11] E. Jones, A. Vedaldi, and S. Soatto. Inertial structure from motion with autocalibration. In *Proceedings of the Internetional Conference on Computer Vision - Workshop on Dynamical Vision*, 2007.

[12] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *Int. Journal of Robotics Research*, 30(1):56–79, 2011.

[13] A. I. Eliazar and R. Parr. Learning probabilistic motion models for mobile robots. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2004.

[14] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[15] C. Gao and J. R. Spletzer. On-line calibration of multiple lidars on a mobile vehicle platform. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[16] J. Underwood, A. Hill, and S. Scheding. Calibration of range sensor pose on mobile platforms. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[17] G. Antonelli, F. Caccavale, F. Grossi, and A. Marino. Simultaneous calibration of odometry and camera for a differential drive mobile robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[18] A. Censi, L. Marchionni, and G. Oriolo. Simultaneous maximum-likelihood calibration of robot and sensor parameters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[19] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

[20] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[21] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[22] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[23] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 2009.

[24] K. Konolige. A gradient method for realtime robot control. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2000.

[25] E. Olson, M. Walter, J. Leonard, and S. Teller. Single cluster graph partitioning for robotics applications. In *Proceedings of Robotics Science and Systems*, pages 265–272, 2005.

[26] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[27] S. Julier. The scaled unscented transformation. In *Proc. of the IEEE Amer. Control Conf*, 2002.

[28] A. Censi, A. Franchi, L. Marchionni, and G. Oriolo. Simultaneous calibration of odometry and sensor parameters for mobile robots, 2012. (preprint, journal version of ICRA paper).

[29] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes, 2nd Edition*. Cambridge Univ. Press, 1992.

[30] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 2011.

[31] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^{2}o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[32] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Journal of Robotics and Autonomous Systems, RAS*, 57:1198–1210, Dec 2009.

[33] E. Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, MIT, Cambridge, MA, USA, June 2008.

[34] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal on Field Robotics*, 23(1):3–20, 2006.