Self-supervised Calibration for Robotic Systems

Jérôme Maye, Paul Furgale, and Roland Siegwart
Autonomous Systems Lab, ETH Zurich, Switzerland
email: {jerome.maye, paul.furgale, roland.siegwart}@mavt.ethz.ch

Abstract—We present a generic algorithm for self calibration of robotic systems that utilizes two key innovations. First, it uses information theoretic measures to automatically identify and store novel measurement sequences. This keeps the computation tractable by discarding redundant information and allows the system to build a sparse but complete calibration dataset from data collected at different times. Second, as the full observability of the calibration parameters may not be guaranteed for an arbitrary measurement sequence, the algorithm detects and locks unobservable directions in parameter space using a truncated QR decomposition of the Gauss-Newton system. The result is an algorithm that listens to an incoming sensor stream, builds a minimal set of data for estimating the calibration parameters, and updates parameters as they become observable, leaving the others locked at their initial guess.

Through an extensive set of simulated and real-world experiments, we demonstrate that our method outperforms state-of-the-art algorithms in terms of stability, accuracy, and computational efficiency.

I. INTRODUCTION

Every robotic system has some set of parameters—scale factors, sensor locations, link lengths, etc.—that are needed for state estimation, planning, and control. Despite best efforts during construction, some parameters will change over the lifetime of a robot due to normal wear and tear. In the best case, incorrect parameter values degrade performance. In the worst case, they cause critical safety issues.

We are interested in developing automated systems that are capable of robust long-term deployment in the hands of non-experts, so the automatic identification and update of these parameter values is highly important.

As an example, consider a camera-based collision avoidance system intended for deployment in a consumer automobile. For the vehicle to be able to avoid collisions, the pose of each camera with respect to the vehicle coordinate system must be known precisely so that obstacle positions can be transformed from camera coordinates into vehicle coordinates. However, a consumer vehicle will have no access to special calibration hardware or expert data analysis. In such a scenario, the vehicle must be capable of self-supervised recalibration. This problem is inherently difficult for a number of reasons that we will briefly discuss here.

1) Parameters change over time—although the vehicle may be factory calibrated, the transformations can change slowly over time due to vibration, thermal expansion, loose parts, or any number of other common problems that follow from normal usage.

2) Parameters must be inferred from the data—as the cameras may be installed in different places on the vehicle, their pose cannot be measured directly. Instead, the transformations must be inferred from the data produced by the full system. However, this is only possible if the motion of the vehicle renders the parameters observable\(^1\).

3) Normal operation may result in unobservable directions in parameter space—unfortunately, for this and many other practical problems normal operation may not render all directions in parameters space observable. In this example, when two cameras do not share an overlapping field of view, planar motion renders the calibration problem degenerate\(^2\); the transformation between cameras only becomes observable under general 3D motion.

4) Unobservable directions in parameter space may appear observable in the presence of noise—even if our hypothetical car is piloted only on a plane (a degenerate case), noise in the measurements can make unobservable parameters appear observable. We call these parameters numerically unobservable to mirror the related concept of numerically rank-deficient matrices.

Existing algorithms for self calibration generally handle issues (1) and (2). Issue (3) is dealt with by designing experiments that guarantee all parameters become observable. To the best of our knowledge, no published self-calibration

\(^1\)Informally, observability means that the parameters can be inferred from some local batch of measurement data [1]

algorithm is able to cope with issue (4). Therefore, unless we plan to require all vehicle owners to outfit their parking places with calibration patterns or regularly drive off-road, new advances for online system calibration are required.

In this paper, we propose an algorithm to deal with all of these difficulties. Our approach exploits the algebraic links between the Gauss-Newton algorithm, the Fisher Information Matrix, and nonlinear observability analysis to automatically detect directions in parameter space that are numerically unobservable and avoid updating our parameters in these directions; at any given time, directions in the parameter space that are observable will be updated based on the latest information, while unobservable directions will remain at their initial guess. Novel sets of measurements are detected using a mutual information test, and added to a working set that is used to estimate the parameters. The result is an algorithm that listens to an incoming stream of data, automatically accumulating a batch of data that may be used to calibrate a full robot system. The only requirements are that (i) the parameters are theoretically observable given some ideal set of data, (ii) it is possible to implement a batch Gauss-Newton estimator for the system state and calibration parameters based on a set of measurements, and (iii) we have some reasonable initial guess for the calibration parameters (e.g. from factory calibration). Fig. 1 shows a typical calibration run of our algorithm.

The remainder of the paper is structured as follows. Section II will give a brief overview over related approaches. Section III is dedicated to the mathematical grounding of our method. Section IV demonstrates the validity of our approach through extensive experiments and evaluation. Section V will conclude the paper.

II. RELATED WORK

The problem of sensor calibration has been a recurring one in the history of robotics and computer vision. Thereby, it has been addressed using a variety of sensor setups and algorithms. A calibration process may involve recovering intrinsic, e.g. focal length for a camera, and extrinsic parameters, i.e. the rigid transformation between the sensor’s coordinate system and a reference coordinate system. For the former, one could devise a naïve approach where the parameters are accurately determined during the manufacturing process. Similarly, the transformation could be retrieved by means of some measuring instrument. However, the disadvantages of such purely engineered methods are manifold. Apart from their impracticality, it can be nearly impossible to reach a satisfying accuracy and thus hinder the proper use of the sensor in a robotic system. Furthermore, external factors such as temperature variations or mechanical shocks may seriously bias a factory calibration. Therefore, much efforts have been dedicated over the years to develop algorithms for calibration of systems in the field.

The use of a known calibration pattern such as a checkerboard coupled with nonlinear regression has become the most popular method in computer vision during the last decade. It has been deployed both for intrinsic camera calibration [4] and extrinsic calibration between heterogeneous sensors [5]. While being relatively efficient, this procedure still requires expert knowledge to reach a good level of accuracy. It can also be quite inconvenient on a mobile platform requiring frequent recalibration.

In an effort to automate the process in the context of mobile robotics, several authors have included the calibration problem in a state-space estimation framework, either with filtering [6] or smoothing [7] techniques. Filtering techniques based on the Kalman filter are appealing due to their inherently online nature. However, in case of nonlinear systems, smoothing techniques based on iterative optimization are usually superior in terms of accuracy. While building on this latter method, our approach attempts to reduce its computational load and copes with degenerate cases.

More recently, Brookshire and Teller [8] have carried out a formal observability analysis in order to identify degenerate paths of the calibration run. In contrast to the method presented in this paper, their approach still expects that the robot travels along non-degenerate paths.

A last class of methods relies on an energy function to be minimized. For instance, Levinson and Thrun [9] have defined an energy function based on surfaces and Sheehan et al. [10] on an information theoretic quantity measuring point cloud quality.

To the best of the authors’ knowledge, little research has been devoted to efficiently deal with degenerate cases frequently occurring during calibration. The majority of the authors indeed assumes that their optimization routine runs on well-behaved data. As demonstrated in the next sections, this can be a critical point in real-world scenarios. In contrast, our approach does not require any of these assumptions and is suitable for online, autonomous, and long-term operations on various platforms and sensors.

III. MODEL

In this section, we will first state our problem in a probabilistic manner and present the mathematical groundings of our method. A practical algorithmic sketch will conclude the presentation.

A. Problem Formulation

In the following, we borrow the formalism of the probabilistic discrete-time Simultaneous Localization and Mapping (SLAM) model [11]. For the sake of clarity, we consider here a robot with a single sensor observing a known number of landmarks at each timestep. Furthermore, we assume the correspondences between sensor’s measurements and landmarks are known. While this latter assumption involves data association techniques which go beyond the scope of this paper, the extension to multiple sensors is straightforward.

Let $X = \{x_{0,K}\}$ be a set of latent random variables (LRV) representing robot states up to timestep $K$, $U = \{u_{1,K}\}$ a set of observable random variables (ORV) representing measured control inputs, $L = \{l_{1,N}\}$ a set of LRV representing $N$ landmark positions, $Z = \{z_{1,N,K,s}\}$ a set of ORV representing $K \times N$ landmark measurements, and $\Theta$ an LRV representing
the calibration parameters of the robot’s sensor. The goal of
the calibration procedure is to compute the posterior margi-
nal distribution of \( \Theta \) given all the measurements up to
timestep \( K \),
\[
p(\Theta \mid U, Z) = \int_{X_L} p(\Theta, X, L \mid U, Z). \tag{1}
\]
Following [11], the full joint posterior on the right-hand side
of (1) may further be factorized into
\[
p(\Theta, X, L \mid U, Z) \propto p(\Theta, X_0, L) \prod_{k=1}^{K} p(x_k \mid x_{k-1}, u_k) \prod_{i=1}^{N} p(z_{k_i} \mid x_i, t_i, \Theta). \tag{2}
\]
We may then approximate (2) with a normal distribution
whose mean, \( \mu_{\Theta,X,L} \), and covariance, \( \Sigma_{\Theta,X,L} \), have to be
estimated. To this end, we first derive a Maximum a Posteriori
(MAP) estimator for the mean,
\[
\hat{\mu}_{\Theta,X,L} = \arg \max_{\Theta,X,L} p(\Theta, X, L \mid U, Z)
\]
\[
= \arg \min_{\Theta,X,L} -\log p(\Theta, X, L \mid U, Z). \tag{3}
\]
When setting the prior \( p(\Theta, X_0, L) \) to a uniform distribu-
tion, (3) becomes a Maximum Likelihood (ML) estimator.

We shall further refine our problem by defining motion and
observation models,
\[
x_k = h(x_{k-1}, u_k, w_k), \]
\[
z_k = g(x_k, t_i, \Theta, n_k), \tag{4}
\]
where
\[
w_k \sim N(0, W_k), \text{ and } n_k \sim N(0, N_k) \tag{5}
\]
are normally distributed process and observation noise vari-
ables, with known covariance \( W_k \) and \( N_k \). Although, the
functions \( h(\cdot) \) and \( g(\cdot) \) might be nonlinear in their para-
eters, we can approximate \( p(x_k \mid x_{k-1}, u_k) \) and \( p(z_k \mid x_k, t_i, \Theta) \) as
normal distributions through linearization.

B. Least Squares Solution

In case of linear motion and observation models, there exists
a closed-form solution to (3) based on the least squares
method due to the normally distributed noise variables. In the
other case, one can resort to nonlinear least squares methods
that iteratively solve a linearized version of the problem. In
the following, we employ the Gauss-Newton algorithm [12]
for this purpose.

From (3) and the approximation that the densities involved
are normal, we can turn the MAP problem into the minimiza-
tion of a sum of squared error terms. Nonlinear optimization
techniques start with an initial guess, \( \hat{\mu}_{\Theta,X,L} \), and refine it
iteratively with \( \hat{\delta \mu}_{\Theta,X,L} \) until convergence. \( \hat{\delta \mu}_{\Theta,X,L} \) is chosen
in such a way that it minimizes a quadratic approximation of
the objective function around \( \hat{\mu}_{\Theta,X,L} \). Gauss-Newton method
only requires the stacked Jacobian matrix of the error terms,
\( H \), in block matrix form, the update takes the form
\[
(H^T T^{-1} H) \hat{\delta \mu}_{\Theta,X,L} = -H^T T^{-1} e(\hat{\mu}_{\Theta,X,L}), \tag{6}
\]
where \( T \), the error covariance matrix, is built from diagonal
blocks of \( W_k \) and \( N_k \), and \( e(\hat{\mu}_{\Theta,X,L}) \) is the error evaluated at
the current estimate \( \hat{\mu}_{\Theta,X,L} \).

At convergence of the algorithm to the estimate \( \hat{\mu}_{\Theta,X,L} \), the
quantity \( H^T T^{-1} H \) is the Fisher Information Matrix (FIM)
and the inverse of the estimate covariance matrix, \( \Sigma_{\Theta,X,L} \).

If we let \( T^{-1} = LL^T \) be the Cholesky decomposition
of the error covariance matrix, (6) can be rewritten as
\[
(LH)^T (LH) \hat{\delta \mu}_{\Theta,X,L} = -(LH)^T L e(\hat{\mu}_{\Theta,X,L}), \tag{7}
\]
which we can recognize as the normal equations of the linear
system
\[
(LH) \hat{\delta \mu}_{\Theta,X,L} = -Le(\hat{\mu}_{\Theta,X,L}). \tag{8}
\]
Thus, instead of solving (6), we can take advantage of matrix
decompositions to solve (8) directly. Not only can this be
more computationally efficient, the use of a rank-revealing
decomposition allows the estimation of the numerical rank
of the FIM, and consequently (as described below) provides
information about the numerical observability of the param-
eters for a given batch of data.

Let \( LH \) be of size \( m \times n \) with the following thin\(^3\) Singular
Value Decomposition (SVD) decomposition [13],
\[
LH = U_n S_n V_n^T, \tag{9}
\]
where \( U_n \) is an \( m \times n \) matrix with orthogonal columns,
\( S_n = \text{diag}(\sigma_1, \cdots, \sigma_n) \), \( V_n \) is an \( n \times n \) matrix with orthogonal
columns, and \( \sigma_i \) are the singular values of \( LH \). From (9) and
using the orthogonality of \( U_n \) and \( V_n \), we can solve (6) as
\[
\hat{\delta \mu}_{\Theta,X,L}^{(\text{SVD})} = -V_n S_n^{-1} U_n^T L e(\hat{\mu}_{\Theta,X,L}). \tag{10}
\]
Although useful for illustrating the concepts of our appro-
ach, the SVD decomposition can be computationally
demanding for large matrices. In practice, we therefore use
the thin \( QR \) decomposition [13] of \( LH \),
\[
LH = Q_n R_n, \tag{11}
\]
where \( \Pi \) is a permutation matrix, \( Q_n \) an \( m \times n \) matrix with
orthogonal columns, and \( R_n \) an \( n \times n \) upper triangular matrix.
In standard QR decomposition, \( \Pi \) is the identity matrix,
otherwise it reflects column pivoting. From (11) and using
the orthogonality of \( Q_n \), (6) can be expressed as
\[
R_n H^T \hat{\delta \mu}_{\Theta,X,L}^{(QR)} = -Q_n^T L e(\hat{\mu}_{\Theta,X,L}), \tag{12}
\]
which, due the upper triangular form of \( R_n \), can be easily
solved by back substitution.

\(^3\)Only the first \( n \) out of \( m \) columns of \( U \) are computed in a thin SVD
decomposition.
The $R_n$ matrix can also be used to compute the FIM and its inverse, the estimate covariance, $\Sigma_{\Theta XL}$. If we drop the permutation matrix for clarity, the FIM simply becomes $R_n^T R_n$ and there exists an efficient algorithm [14] for recovering any elements of the covariance estimate without inverting the whole FIM.

At the convergence of the Gauss-Newton optimization, we are thus left with the estimates $\hat{\mu}_{\Theta XL}$ and $\hat{\Sigma}_{\Theta XL}$ of the normal distribution $p(\Theta, X, L | U, Z)$. In order to solve (1), we can employ the marginalization property of normal distributions [15]. If we express the estimates in the partitioned form,

$$
\begin{pmatrix}
\hat{\mu}_X \\
\hat{\mu}_L \\
\hat{\mu}_\Theta
\end{pmatrix}, \quad
\begin{pmatrix}
\hat{\Sigma}_{XX} & \hat{\Sigma}_{XL} & \hat{\Sigma}_{X\Theta} \\
\hat{\Sigma}_{LX} & \hat{\Sigma}_{LL} & \hat{\Sigma}_{L\Theta} \\
\hat{\Sigma}_{\Theta X} & \hat{\Sigma}_{\Theta L} & \hat{\Sigma}_{\Theta\Theta}
\end{pmatrix},
$$

then $p(\Theta | U, Z) \sim N(\hat{\mu}_\Theta, \hat{\Sigma}_{\Theta})$. We can thus extract $\hat{\mu}_\Theta$ from $\hat{\Sigma}_{\Theta XL}$ very efficiently, and by placing $\Theta$ to the right of the Jacobian matrix $H$ and using [14], $\hat{\Sigma}_{\Theta}$ can be computed in $O(l)$, where $l = \text{card}(\Theta)$.

C. Truncated SVD and QR Solutions

There exists a solution to (6) iff the FIM is invertible, i.e., it is full rank. Jauffret [16] clearly articulates the link between the rank of the FIM and observability of the parameters being estimated. In the context of calibration, a singular FIM corresponds to some unobservable directions in the parameter space given the current set of observations. Classical observability analysis, for example the widely used method of Hermann and Krener [17] (c.f. [18] or [19]), proves what we will call structural observability—that there exists some dataset for which the parameters are observable—it does not guarantee that the parameters are observable for any dataset.

In real-world scenarios, where data is contaminated by noise, the FIM might appear to be full rank although it is actually rank-deficient. We can illustrate this with the matrix

$$
A = \begin{pmatrix}
0.9999 & 1.9999 & 3.0014 \\
4.0007 & 5.0015 & 6.0007 \\
6.9998 & 8.0014 & 8.9998
\end{pmatrix}.
$$

Although $A$ is technically full rank, it was constructed from a structurally rank-deficient matrix (the second row is a linear combination of the first and the last row) with added Gaussian noise.

Using SVD decomposition, we can identify a numerically rank-deficient matrix by analyzing its singular values [20] and consequently the numerical observability of the system [21]. The numerical rank $r$ of a matrix is defined as the index of its smallest singular value $\sigma_r$ larger than a user-defined tolerance $\epsilon$, i.e.,

$$
r = \arg \max_i \sigma_i \geq \epsilon.
$$

For example, the singular values of $A$ are approximately 16.8, 1.1, and 0.0001. From (10), we can see that if some of the $\sigma_i$ are close to zero, i.e. $r < n$, the update vector $\delta p^{(TSVD)}_{\Theta XL}$ will be large (scaled by $1/\sigma_i$) and sometimes lead to a divergence of the solution. In order to cope with this issue, we can approximate $LH$ with a lower-rank matrix yielding the Truncated SVD (TSVD) [23] solution,

$$
\delta p^{TSVD}_{\Theta XL} = -V_r S_r^{-1} U_r^T L e \hat{\mu}_{\Theta XL},
$$

with $U_r$ an $m \times r$ matrix, $S_r = \text{diag}(\sigma_1, \cdots, \sigma_r)$, $V_r$ an $n \times r$ matrix.

It is worth noting that TSVD applies a sharp filter to the system, whereas regularization methods such as Tikhonov or ridge regression work as a smooth filter. As shown in [24], it nevertheless produces similar results at lower computational costs when the matrix has a well-determined numerical rank, i.e., a clearly defined jump in the singular values spectrum.

A similar approach can be derived using rank-revealing QR decomposition [25] yielding the Truncated QR (TQR) method [26]. The strategy is to apply QR factorization with column pivoting in order to reveal the rank and to apply a cutoff on the diagonal of the R matrix. It can further be noted that the Q matrix need not be explicitly constructed, but instead returned in the form of Householder reflections that can be applied to the right-hand side of the linear system.

According to the discussion in [13], column scaling may improve the conditioning of the system, i.e. the ratio between the largest and the smallest singular value, and hence the stability of the solution. Within our framework, scaling is also relevant for merging quantities with miscellaneous magnitudes and helps in setting a rank tolerance $\epsilon$ that is widely applicable. We define the scaling matrix as

$$
G = \text{diag}\left(\frac{1}{||LH(:,1)||}, \cdots, \frac{1}{||LH(:,m)||}\right),
$$

where $|| \cdot ||$ denotes the column vector norm and $H(:,i)$ the $i$-th column of $H$, and compute a solution to the scaled system,

$$
LHG \delta p^{(\epsilon)}_{\Theta XL} = -L e \hat{\mu}_{\Theta XL}.
$$

The unscaled solution is finally expressed as $G \delta p^{(\epsilon)}_{\Theta XL}$.

D. Selecting Informative Measurements

Since we are mainly interested in calibrating our sensor and thus in the posterior $p(\Theta | U, Z)$, we do not need to consider all the measurements. Let us define a partition of the measurements $D_1 = \{u_{1:i}, z_{1:i}\}$ and $D_2 = \{u_{i+1:K}, z_{i+1:K}\}$ with $i < K$. Using information theory, we can quantify the information gain when estimating $\Theta$ with $D_1$ alone or with $D_1 \cup D_2$.

The mutual information (MI) [27] between two random variables $X$ and $Y$ is defined as

$$
I(X; Y) = \int_X \int_Y p(x, y) \log \frac{p(x, y)}{p(x)p(y)},
$$

where $p(x, y)$ is the joint density of $X$ and $Y$, and $p(x)$ and $p(y)$ their marginal densities. The MI measures the amount
of information \( X \) and \( Y \) share or, in other words, quantifies the reduction of uncertainty in \( X \) when knowing \( Y \).

Before deriving the MI for our purpose, it is worth recalling another property of normal distributions [15]. If we define \( \Theta_1 = \Theta \mid D_1 \) and consider the joint normal distribution \( p(\Theta_1, D_2) \) with parameters

\[
\mu_{\Theta_1; D_2} = \left( \mu_{\Theta_1}, \mu_{D_2} \right), \quad \Sigma_{\Theta_1; D_2} = \begin{pmatrix} \Sigma_{\Theta_1; \Theta_1} & \Sigma_{\Theta_1; D_2} \\ \Sigma_{D_2; \Theta_1} & \Sigma_{D_2; D_2} \end{pmatrix}
\]

then \( p(\Theta_1 \mid D_2) \) is a normal distribution with parameters

\[
\mu_{\Theta_1; D_2} = \mu_{\Theta_1} + \Sigma_{\Theta_1; D_2} \Sigma_{D_2; D_2}^{-1} (D_2 - \mu_{D_2}) \\
\Sigma_{\Theta_1; D_2} = \Sigma_{\Theta_1; \Theta_1} - \Sigma_{\Theta_1; D_2} \Sigma_{D_2; D_2}^{-1} \Sigma_{D_2; \Theta_1}.
\]

Furthermore, \( \Sigma_{\Theta_1; D_2} \) can be recognized as the Schur complement of the joint covariance matrix \( \Sigma_{\Theta_1; D_2} \).

Following an argument of [28], the MI between \( \Theta_1 \) and \( D_2 \) can then be computed as

\[
I(\Theta_1; D_2) = \frac{1}{2} \log \left| \frac{\Sigma_{\Theta_1; \Theta_1}}{\Sigma_{\Theta_1; D_2} \Sigma_{D_2; \Theta_1}} \right| = \frac{1}{2} \log \left| \Sigma_{\Theta_1; \Theta_1} \right| - \frac{1}{2} \log \left| \Sigma_{\Theta_1; D_2} \right|.
\]

where \( | \cdot | \) denotes the matrix determinant and \( \Sigma_{\Theta_1; D_2} \) is the covariance estimate of \( \Theta_1 \) using \( D_1 \) and \( D_2 \), which is computed by Gauss-Newton optimization. Thus, using (22), we can measure the amount of information \( D_2 \) conveys to our current estimate \( \Theta_1 \).

E. Algorithm

Thus far, we have delivered a formal introduction to the probabilistic groundings of our self-supervised calibration approach. This section will be dedicated to a practical online implementation of our algorithm.

The proposed calibration method is sketched in Alg. 1. Defining \( D^{info} \) as the set of informative measurements at time \( t \), we collect a measurement batch \( D^{new} = \{u_{t+k}, z_{n, t+k}, y \} \) during \( k \) timesteps and compute \( p(\Theta \mid D^{info}, D^{new}) \) using Gauss-Newton method with TQR updates and threshold \( \epsilon \). In a second step, if \( I(\Theta \mid D^{info}, D^{new}) > \lambda \) then the estimate of \( \Theta \) is updated.

Given the structure of the model (2), the Jacobian matrix \( H \) will be sparsely populated. Therefore, in order to obtain a tractable and scalable method, we adopt sparse matrices algorithms and data structures. The memory and computational costs are then \( O(\text{card}(X) \times \text{card}(L)) \).

IV. Experiments

In order to evaluate and validate the approach proposed in this paper, we have conducted experiments on simulated and real-world data. We shall first start with the description of the experimental conditions and then demonstrate the performance of our algorithm, along with some comparisons against existing methods.

A. Experimental Setup

Our method has been fully implemented in MATLAB and we have used the SuiteSparseQR [29] package for performing sparse matrix operations.

Although our eventual goal is full, self-supervised calibration for an autonomous vehicle system with multiple heterogeneous sensors, the experiments in this paper are based on a realistic, but somewhat simplified robotic system. This allows us to precisely control the observability properties in simulation and test the behavior of our algorithm.

Fig. 2 depicts our experimental setup. The platform is a differential drive mobile robot equipped with a range sensor delivering range and bearing measurements. The calibration parameters of the range sensor consist in the transformation of its coordinate system to the robot’s coordinate system. The platform is further endowed with wheel odometers outputting translational and rotational speeds. While navigating on the plane, the robot observes a known number of landmarks through its range sensor. Both the range sensor and the odometry produce data at 10 Hz.

More formally, we adopt the following motion and observation models

```plaintext
Algorithm 1: calibrateSensor()

Input: Initial guesses \( \Theta^{(0)}, \hat{x}^{(0)}, \hat{z}^{(0)} \) models
Input: Motion \( h(\cdot) \) and observation \( g(\cdot) \) models
Input: Batch size \( k \), TQR threshold \( \epsilon \), MI threshold \( \lambda \)
Output: \( \hat{\mu}_\Theta \) and \( \hat{\Sigma}_\Theta \)

\[ D^{info} \leftarrow \emptyset; \]
\( t \leftarrow 1; \)

while calibrate do

// Collecting measurements:
\( t^{\text{init}} \leftarrow t; \)
\( D^{new} \leftarrow \emptyset; \)
while \( t < t^{\text{init}} + k \) do

\( \hat{x}^{(t)} \leftarrow h(x^{(t)}, u, \theta); \)
\( D^{new} \leftarrow D^{new} \cup \{u, z_{t+k}\}; \)
\( t \leftarrow t + 1; \)
end

// TQR optimization with threshold \( \epsilon \):
\( \hat{\mu}_{\Theta, X, L} \leftarrow \arg \max \Theta, X, L p(\Theta, X, L \mid D^{info}, D^{new}); \)
// Marginalization:
\( \hat{\mu}_\Theta \leftarrow \hat{\mu}_{\Theta, X, L}; \)
\( \hat{\Sigma}_\Theta \leftarrow \hat{\Sigma}_{\Theta, X, L}; \)
// MI decision:
if \( I(\Theta \mid D^{info}, D^{new}) > \lambda \) then

\( D^{info} \leftarrow D^{info} \cup D^{new}; \)
\( \Theta \leftarrow N(\hat{\mu}_\Theta, \hat{\Sigma}_\Theta); \)
end
end
```
Fig. 2. Experimental setup. A 2D robot moving in a plane and observing landmarks with a range sensor providing range and bearing angle measurements. The calibration process needs to find the 2D transformation between the robot’s coordinate system and the sensor’s coordinate system.

\[
\begin{pmatrix}
  x_k \\
y_k \\
\theta_k
\end{pmatrix} = \begin{pmatrix}
x_{k-1} \\
y_{k-1} \\
\theta_{k-1}
\end{pmatrix} + T \begin{pmatrix}
\cos \theta_{k-1} \\
\sin \theta_{k-1} \\
0
\end{pmatrix} \begin{pmatrix}
v_k \\
w_k
\end{pmatrix} + \begin{pmatrix}
0 \\
0 \\
1
\end{pmatrix}
\]

\[
a = x_i - x_k - \delta_x \cos \theta_k + \delta_y \sin \theta_k
\]

\[
b = y_i - y_k - \delta_x \sin \theta_k - \delta_y \cos \theta_k
\]

\[
\begin{pmatrix}
\phi^i_k \\
\phi^\psi_k
\end{pmatrix} = \begin{pmatrix}
\sqrt{a^2 + b^2} \\
\arctan2(b, a) - \theta_k - \psi + \pi/4
\end{pmatrix} + n_k,
\]

where \( x_k = [x_k, y_k, \theta_k]^T \) denotes the robot pose at timestep \( k \), \( T \) the sampling period, \( u_k = [v_k, w_k]^T \) the measured translational and rotational speeds, \( z_k = [\phi^i_k, \phi^\psi_k]^T \) the range and bearing observation of landmark \( i \) with pose \( \ell_i = [x_i, y_i]^T \), \( w_k \sim \mathcal{N}(0, W_k) \) with \( W_k = \text{diag}(\sigma^2_v, \sigma^2_w) \), \( n_k \sim \mathcal{N}(0, N_k) \) with \( N_k = \text{diag}(\sigma^2_\phi, \sigma^2_\theta) \), and \( \Theta = [\delta_x, \delta_y, \psi]^T \) the range sensor’s calibration parameters.

Throughout our experiments, we have used a non-informative prior \( p(\Theta, x_0, L) \), i.e., a uniform distribution. The use of priors is still subject to controversial discussions [30] between Bayesian and non-Bayesian statisticians. Here, we argue that everything should come from the data itself and not from some subjective prior information that could bias the inference.

Our algorithm requires only 3 free parameters, namely the rank threshold \( \epsilon \), the batch size \( k \), and the MI threshold \( \lambda \). Optimally, \( k \) should be inferred from the dynamics of the system. While a large \( k \) induces storage of uninformative measurements, a small \( k \) leads to useless runs of optimization and makes it difficult to discover informative sequences of measurements. In this setup, we have used a batch size of \( k = 100 \) (ten seconds of data). Concerning the MI threshold, a small \( \lambda \) will keep most of the measurements and a large \( \lambda \) will ignore them all. We have set this value to \( \lambda = 0.5 \) [bits] in our experiments. The \( \epsilon \) parameter shall be discussed below.

**B. Simulated Data**

In our simulation environment, we can generate various paths for the robot, along with corresponding sensor measurements, and thus analyze the behavior of multiple algorithms, especially in degenerate cases. We have created an environment with \( N = 17 \) landmarks uniformly distributed on a 20m x 20m grid. We have set the noise parameters empirically to \( \sigma^2_v = 4.4 \times 10^{-3}, \sigma^2_w = 8.2 \times 10^{-2}, \sigma^2_\phi = 9.0036 \times 10^{-4}, \) and \( \sigma^2_\theta = 6.7143 \times 10^{-4} \). The calibration parameters are fixed at \( \delta_x = 0.219 \) [m], \( \delta_y = 0.1 \) [m], and \( \psi = \pi/4 \) [rad].

In a first effort, we want to support the claims of Sec. III-C with a representative example. We simulated the robot driving along a straight path as shown in Fig. 3. Intuitively, the problem is structurally unobservable. Indeed, it has 5 unobservable parameters, 3 corresponding to the global pose of the map and trajectory (as no global measurements or prior are included) and 2 for the calibration offset variables \( \delta_x \) and \( \delta_y \). In the absence of noise, 5 of the singular values of the Jacobian matrix are zero up to machine precision, i.e., a structural rank deficiency. However, when noise is added, only 3 singular values remain at 0 for the same problem. From the integrated odometry path in Fig. 3, the calibration parameters appear indeed as observable and a naive algorithm without regularization will thus wrongly optimize. Fig. 4 shows the singular values in these two cases and the related concept in the QR decomposition. With our TSV/TQR method, we can deal with this issue by setting an adequate \( \epsilon \) threshold that will recover the correct rank deficiency and therefore only optimize the observable parts. The threshold is application-specific and should be a function of the system noise. Obviously, at a certain level of noise, gaps in the singular values spectrum become indistinguishable. In our setup, we have determined the threshold empirically from the scaled system (18) and set it to \( \epsilon = 0.013 \).

To conclude this section on simulated data, we will analyze the performance of different algorithms, namely an Extended
Kalman Filter (EKF) similar to [6], a standard batch nonlinear least squares (LS) method without regularization [7], and our TQR-MI approach, on a sine wave path with varying amplitude such that the offset parameters range from unobservable (straight path or zero amplitude) to fully observable (higher amplitude). To get significant statistical results, we have repeated the experiment 100 times with the same initial conditions ($\delta_0 = 0.23$ [m], $\psi_0 = 0.11$ [m], $\psi_0 = 0.8$ [rad]) and amplitudes from 0 [m] to 5 [m] with steps of 0.5 [m] over 5000 timesteps (500 [s] of data). Fig. 5 displays the result of this experiment. At amplitude 0, we clearly witness that the LS and EKF estimators fail to yield reasonable offset parameters. With our TQR-MI method, these parameters remain at their initial guess until an amplitude of 1 and only the angle gets optimized. As the amplitude grows, our method becomes comparable to the LS estimator, while using fewer measurements. In these experiments, the EKF is fed by a vague initial prior and exhibits instability in case of unobservability. A more concentrated prior would improve the results at the cost of introducing potential bias.

C. Real-world Data

In order to validate our method on real-world data, we have used the “Lost in the Woods Dataset” provided with the courtesy of Tim Barfoot [31]. This dataset contains approximately 20 minutes of a robot driving amongst a forest of tubes which serve as landmarks. The ground truth comes from a motion capture system that tracks robot motion and tube locations. For the calibration parameters, we have only access to $\delta_x = 0.219$ [m] that was roughly measured with a tape. We assume the others are implicitly set to 0 ($\delta_y = 0$ [m] and $\psi = 0$ [rad]). Fig. 6 displays the qualitative result of our algorithm. Using only 10% of the measurements, we could recover accurate landmark positions, robot poses, and calibration parameters. For visualization, estimated landmarks and poses have been aligned to the ground truth using the algorithm from [32].

Since the ground truth calibration parameters are inaccurate, Tab. I compares our method against a standard least squares (LS) estimator that exploits all the measurements.

Our TQR-MI algorithm performs the optimization with $K = 1209$ out of 12609 timesteps. Although the calibration parameters are the same magnitude, the higher variances stem from the fewer number of considered measurements.

V. Conclusion

In this paper, we have presented a novel approach to the automatic calibration of mobile robot sensors. We have included the calibration parameters in a SLAM formulation and computed an MAP estimator using a Gauss-Newton algorithm. To cope with unobservability, we have employed truncated QR decomposition as a regularization method. For long-term and online operations, we have devised a mutual information selection scheme that solely captures informative measurements relevant to the calibration. Our algorithm has been thoroughly tested and validated through extensive simulated and real-world experiments on a 2D robot equipped with a laser range finder (LRF).

In the near future, we will consider the self calibration of multiple sensors (cameras, 3D/2D LRF, GPS/INS system, ...) mounted on an autonomous car and a C++ implementation. From a research point of view, we would like to further automate the selection of the free parameters. Lastly, we want to detect stepwise changes in the calibration parameters in order to trigger recalibration in case of mechanical shocks.
This work has partly been supported by the EC under FP7-269916-V-Charge.

REFERENCES