Model Free Calibration of Wheeled Robots Using Gaussian Process

Mohan Krishna Nutalapati, Lavish Arora, Anway Bose, Ketan Rajawat, and Rajesh M Hegde

Abstract— **Robotic calibration allows for the fusion of data from multiple sensors such as odometers, cameras, etc., by providing appropriate relationships between the corresponding reference frames. For wheeled robots equipped with camera/lidar along with wheel encoders, calibration entails learning the motion model of the sensor or the robot in terms of the data from the encoders and generally carried out before performing tasks such as simultaneous localization and mapping (SLAM). This work puts forward a novel Gaussian Process-based non-parametric approach for calibrating wheeled robots with arbitrary or unknown drive configurations. The procedure is more general as it learns the entire sensor/robot motion model in terms of odometry measurements. Different from existing non-parametric approaches, our method relies on measurements from the onboard sensors and hence does not require the ground truth information from external motion capture systems. Alternatively, we propose a computationally efficient approach that relies on the linear approximation of the sensor motion model. Finally, we perform experiments to calibrate robots with un-modelled effects to demonstrate the accuracy, usefulness, and flexibility of the proposed approach.**

I. INTRODUCTION

Robotic calibration is an essential first step necessary for carrying out various sophisticated tasks such as simultaneous localization and mapping (SLAM) [1], [2], object detection and tracking [3], and autonomous navigation [4]. For most wheeled robot configurations equipped with wheel encoders and exteroceptive sensors like camera/lidar, the calibration process entails learning a mathematical model that can be used to fuse odometry and sensor data. In the case when the motion model of the robot is unavailable, due to some unmodelled effects calibration involves learning the relationships that describe the sensor motion in terms of the odometry measurements. Precise calibration is imperative since calibration errors are often systematic and tend to accumulate over time [5]. Conversely, an accurately specified odometric model complements the exteroceptive sensor, e.g. to correct for measurement distortions if any [6], and continues to provide motion information even in featureless or geometrically degenerate environments [7].

Traditional approaches [8], [9] for calibration of wheeled robots focus on learning a parametric motion model of the robot/sensor. A common issue among these approaches was the need for external measurement setup such as calibrated video cameras or motion capture systems. On the other hand [10], [11] overcome this issue by performing simultaneous calibration of odometry and sensor parameters using measurements from the sensor. More generic calibration routines for arbitrary robot configurations were presented in [12], [13] where solution to calibration parameters is found along with robot state variables. All these techniques essentially

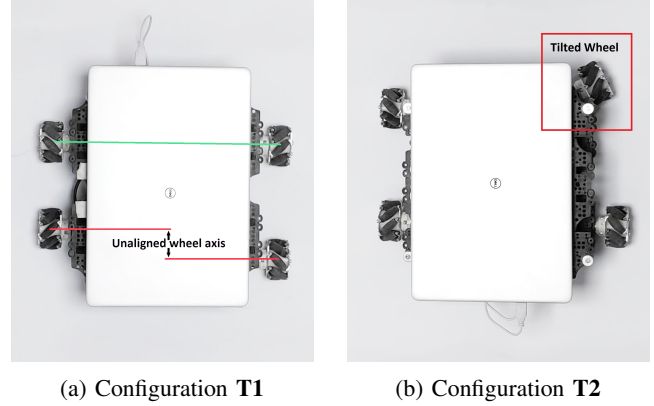(a) Configuration **T1**   (b) Configuration **T2**

Fig. 1: Deformed Turtlebot3 mecanum drive robot used for experimental evaluations. (a) Unaligned wheel axis deformation, (b) Tilted wheel deformation.

learn the parameters associated with the motion model of the robot/sensor to generate accurate odometry. However, the performance degrades due to uncertainty arising from interactions with the ground and hence lead to bad odometry estimates [5]. Moreover, modeling such uncertainties that arise due to non-systematic errors is a very challenging and difficult task. To this end, non-parametric methods [14], [5] employ tools from Gaussian process (GP) estimation learn the residuals between the parametric model and ground truth measurements from external motion capture systems. A common assumption that all these methods make is the residual function being zero mean, which holds true only when the kinematic model of the robot is accurately known. However, for robots with misaligned wheel axis or other unknown offsets that may arise due to unsupervised assembly [15], excessive wear-and-tear etc., zero mean assumption is not valid rendering the approach suboptimal.

This work puts forth a more general framework that subsumes existing non-parametric approaches, while also applicable to scenarios where the motion model of the robot/sensor is distorted or not known. Different from the existing non-parametric approaches, the proposed method learns the whole sensor/robot motion model. To this end, the key contributions of the work are

- Formulation of a Gaussian-process regression framework that captures the arbitrary or unknown motion model of the sensor/robot. The entire calibration routine can be carried out using measurements from onboard sensors capable of sensing ego-motion
- A computationally efficient approach for near-optimal and model-free calibration

The rest of the paper is organized as follows. Sec. II details

TABLE I: Nomenclature used in the paper

Parameters	
$\boldsymbol{p} = (\underbrace{\ell_x, \ell_y, \ell_\theta}_{\boldsymbol{\ell}}, \mathbf{r})$	parameters to be estimated
$\boldsymbol{\ell}$	position of extrinsic sensor w.r.t robot frame
$\mathbf{r}$	robot instrinsic parameters

Measurements		
$\mathcal{U}$		raw data log of odometry sensor
$\mathcal{V}$		Measurememts from exteroceptive sensor
$\mathbf{q}(t)$	$=$	$[q_x(t)\ q_y(t)\ q_\theta(t)]^T$ Pose of robot at any time t
$\hat{\mathbf{s}}_{jk}$		sensor displacement estimate for time interval $[t_j, t_k)$

More Symbols	
$\oplus$	Roto-translation operator
$\ominus$	inverse of $\oplus$ operator
$\begin{bmatrix} a_x \\ a_y \\ a_\theta \end{bmatrix} \oplus \begin{bmatrix} b_x \\ b_y \\ b_\theta \end{bmatrix} \triangleq \begin{bmatrix} a_x + b_x \cos a_\theta - b_y \sin a_\theta \\ a_y + b_x \sin a_\theta + b_y \cos a_\theta \\ a_\theta + b_\theta \end{bmatrix}$	
$\ominus \begin{bmatrix} a_x \\ a_y \\ a_\theta \end{bmatrix} \triangleq \begin{bmatrix} -a_x \cos a_\theta - a_y \sin a_\theta \\ a_x \sin a_\theta - a_y \cos a_\theta \\ -a_\theta \end{bmatrix}$	

Fig. 2: Block diagram describing the system setup and data flow

the system setup and the problem formulation. The proposed algorithm is described in Sec. III. Detailed experimental evaluations are carried out to validate the performance of the proposed method, and the results are discussed in Sec. IV. Finally, Sec. V concludes the paper. The notation used in the paper is summarized in Table I.

II. PROBLEM FORMULATION

In this section we first start with introducing preliminary notations (see Table I) used through out the paper. Consider a general robot with an arbitrary drive configuration, equipped with m rotary encoders on its wheels and/or joints and an exteroceptive sensor such as a lidar or a camera. The exteroceptive sensor can sense the environment and generate scans or images $\mathcal{V} = \{\mathcal{Z}(t)\}_{t \in \mathcal{T}}$ that can be used to estimate its ego motion. Here, $\mathcal{T} := \{t_1, t_2, \dots, t_n\}$ denotes the set of discrete time instants at which the measurements are made. The rotary encoders output raw odometry data in the form of a sequence of wheels angular velocities $\mathcal{U} = \{\boldsymbol{\delta}(t)\}_{t \in \mathcal{T}}$. Given two time instants t_j and t_k such that $\Delta t_{jk} := t_k - t_j > 0$ is sufficiently small, it is generally assumed that $\boldsymbol{\delta}(t) := \boldsymbol{\delta}_{jk}$ for all $t_j \le t < t_k$. Traditionally, the odometry data is pre-processed to yield relative translation motion and orientation information, and is subsequently fused with the ego motion estimates from exteroceptive sensors. This pre-processing step necessitates the use of the motion model $\boldsymbol{f}_r$ of the robot that acts upon the odometry data $\boldsymbol{\delta}_{jk}$ to yield the relative pose of the robot $\mathbf{q}_{jk} := \ominus \mathbf{q}_j \oplus \mathbf{q}_k = \boldsymbol{f}_r(\boldsymbol{\delta}_{jk})$ for the time interval Δt_{jk}. Here $\mathbf{q}_j := (q_j^x, q_j^y, q_j^\theta)^\top$ denote the position of the robot at time $t = t_j$. Note that if the exteroceptive sensor is mounted exactly on the robot frame of reference, the sensor motion model denoted by $\boldsymbol{f}$ is the same as the robot motion model $\boldsymbol{f}_r$. In general however, if the pose of the exteroceptive sensor with respect to the robot is denoted by $\boldsymbol{\ell}$, the sensor motion model is given by $\boldsymbol{f}(\boldsymbol{\delta}_{jk}) = \ominus \boldsymbol{\ell} \oplus \boldsymbol{f}_r(\boldsymbol{\delta}_{jk}) \oplus \boldsymbol{\ell}$, where generally $\boldsymbol{\ell}$ is unknown.

Having the preliminary notations at hand, we now describe the system setup displayed in Fig. 2. The goal of the calibra-
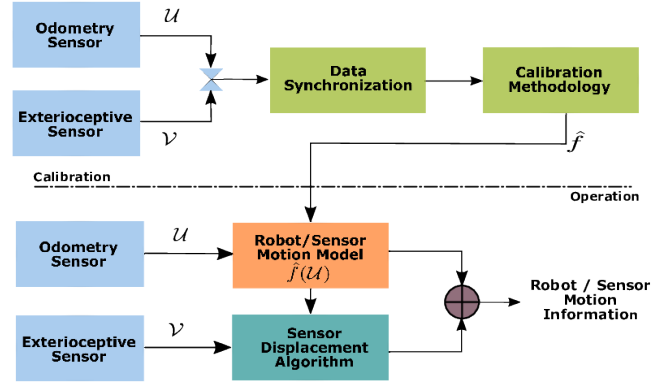
tion phase is to estimate the function $\boldsymbol{f}$, given $\mathcal{U}$ and $\mathcal{V}$. The estimated motion model, denoted by $\hat{\boldsymbol{f}}$, is subsequently used in the operational phase to augment or even complement the motion estimates provided by the exteroceptive sensor. For instance, accurate odometry can be used to correct distortions in the sensor measurements [6]. Note here that the parametric form of the function $\boldsymbol{f}$ exists when the robot motion model $\boldsymbol{f}_r$ is well defined. For instance two-wheel differential drive robot [10] , four wheel mecanum drive [16] etc. In other words, $\boldsymbol{f}(\bullet) = \boldsymbol{g}(\bullet ; \boldsymbol{p})$ where $\boldsymbol{g}$ is a known function and $\boldsymbol{p}$ is the set of unknown parameters, such as the dimensions of the wheel, sensor position w.r.t robot frame of reference etc. State-of-the-art techniques like [10], [11], [17] learns $\boldsymbol{f}$ under this assumption. A significantly more challenging scenario occurs when the form of $\boldsymbol{f}$ is not known, e.g. due to excess wear-and-tear, or is difficult to handle, e.g. due to non-differentiability. For such cases, the parametric approaches [10], [11], [17] are no longer feasible and the unknown function $\boldsymbol{f}$ is generally infinite-dimensional. Towards this end, a low-complexity approach is proposed (see Sec. III-B), wherein a simple but generic (e.g. linear) model for $\boldsymbol{f}$ is postulated. A more general and fully non-parametric Gaussian process framework is also put forth that is capable of handling more complex scenarios and estimate a broader class of motion models $\boldsymbol{f}$. It is remarked that in this case, unless the exteroceptive sensor is mounted on the robot axis, additional information may be required to also estimate the robot motion model $\boldsymbol{f}_r$.

III. MODEL FREE CALIBRATION USING GP

When no information about the kinematic model of the robot is available, it becomes necessary to estimate $\boldsymbol{f}$ directly. As in Sec. II, let $\mathcal{T}$ be the set of time instants at which measurements are made. For certain time interval $[t_j, t_k)$ for which Δt_{jk} is not too large, let the exteroceptive sensor generates motion estimates $\hat{\mathbf{s}}_{jk}$. Given data of the form $\mathcal{D} := (\boldsymbol{\delta}_{jk}, \hat{\mathbf{s}}_{jk})_{(j,k) \in \mathcal{E}}$, where $\mathcal{E}$ represents set of indices of all chosen key measurement pairs of the sensor and $n := |\mathcal{E}|$, the goal is to learn the function $\boldsymbol{f} : \mathbb{R}^m \to \mathbb{R}^3$ that adheres to the model

$$\hat{\mathbf{s}}_{jk} = \boldsymbol{f}(\boldsymbol{\delta}_{jk}) + \boldsymbol{\varepsilon}_{jk} \qquad (1)$$

for all $(j, k) \in \mathcal{E}$, where $\varepsilon_{jk} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{jk}) \in \mathbb{R}^3$ models the noise in the measurements, and $\boldsymbol{\delta}_{jk}$ now represents the number of wheel ticks recorded in the time interval Δt_{jk}. Here we assume that the noise covariance $\boldsymbol{\Sigma}_{jk}$ is known before hand. Given an estimated $\hat{\boldsymbol{f}}$ of the sensor motion model, a new odometry measurements $\boldsymbol{\delta}_e$ for an edge e (pair of times at which measurements are made) can be used to directly yield sensor pose changes $\mathbf{s}_e = \hat{\boldsymbol{f}}(\boldsymbol{\delta}_e)$. As remarked earlier, it may be possible to obtain the robot pose change $\mathbf{q}_e$ from $\mathbf{s}_e$ if the sensor pose ℓ is known a priori. Since the functional variable $\boldsymbol{f}$ is infinite dimensional in general, it is necessary to postulate a finite dimensional model that is computationally tractable. Towards solving the functional estimation problem, we detail two methods, that are very different in terms of computational complexity and usage flexibility.

A. Calibration via Gaussian process regression (CGP)

The GP regression approach assumes that the measurement is Gaussian distributed and that the function $\boldsymbol{f}$ is a Gaussian process, whose mean and variance functions depend on the data. Specifically, we have that

$$\hat{\mathbf{s}}_{jk} \sim \mathcal{N}(\boldsymbol{f}(\boldsymbol{\delta}_{jk}), \boldsymbol{\Sigma}_{jk}) \tag{2}$$

or equivalently, $\varepsilon_{jk} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{jk})$. Given inputs $\{\boldsymbol{\delta}_{jk}\}$, let $\mathbf{f}$ denote the $\{3n \times 1\}$ vector that collects $\{\boldsymbol{f}(\boldsymbol{\delta}_{jk})\}$ for $\{(j, k) \in \mathcal{E}\}$. Defining $\boldsymbol{\Sigma} \in \mathbb{R}^{3n \times 3n}$ as the block diagonal matrix with entries $\boldsymbol{\Sigma}_{jk}$ and $\hat{\mathbf{s}} \in \mathbb{R}^{3n}$ as the vector that collects all the measurements $\{\hat{\mathbf{s}}_{jk}\}_{(j,k) \in \mathcal{E}}$. Having this we can equivalently write the joint likelihood as

$$p(\hat{\mathbf{s}}|\mathbf{f}) = \mathcal{N}(\hat{\mathbf{s}}|\mathbf{f}, \boldsymbol{\Sigma}) \tag{3}$$

Unlike the parametric model based approaches [10], we impose a Gaussian process prior on $\boldsymbol{f}$ directly. Equivalently, we have that

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\bar{\boldsymbol{\mu}}, \mathbf{K}) \tag{4}$$

where $\bar{\boldsymbol{\mu}} \in \mathbb{R}^{3n}$ is the mean vector with stacked entries of $\boldsymbol{\mu}(\boldsymbol{\delta}_{jk}) \in \mathbb{R}^3$, and $\mathbf{K} \in \mathbb{R}^{3n \times 3n}$ is the covariance matrix with a block of entries $[\mathbf{K}_{i,i'}] = \boldsymbol{\kappa}(\boldsymbol{\delta}_{jk}, \boldsymbol{\delta}_{j'k'})$ for (j, k) and $(j', k') \in \mathcal{E}$ and $i, i' \in \{1, \cdots, n\}$. The choice of the mean function $\boldsymbol{\mu} : \mathbb{R}^m \to \mathbb{R}^3$ and the kernel function $\boldsymbol{\kappa} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{3 \times 3}$ is generally important and application specific. Popular choices include the linear, squared exponential, polynomial, Laplace, and Gaussian, among others. With a Gaussian prior and noise model, the posterior distribution of $\boldsymbol{f}$ given $\mathcal{D}$ is also Gaussian. For a new odometry measurement $\boldsymbol{\delta}_e$ with noise variance $\boldsymbol{\Sigma}_e$, let $\mathbf{k}_e \in \mathbb{R}^{3n \times 3}$ be the vector that collects $\{\boldsymbol{\kappa}(\boldsymbol{\delta}_e, \boldsymbol{\delta}_{jk})\}_{(j,k) \in \mathcal{E}}$. Then the distribution of $\hat{\boldsymbol{f}}(\boldsymbol{\delta}_e)$ for given $\hat{\mathbf{s}}$ is

$$p(\hat{\boldsymbol{f}}(\boldsymbol{\delta}_e)|\hat{\mathbf{s}}) = \mathcal{N}(\hat{\boldsymbol{f}}(\boldsymbol{\delta}_e) \mid \hat{\boldsymbol{\mu}}_e, \hat{\boldsymbol{\Sigma}}_e) \tag{5}$$

where $\hat{\boldsymbol{\mu}}_e = \mathbf{k}_e^T(\mathbf{K}+\boldsymbol{\Sigma})^{-1}(\hat{\mathbf{s}}-\bar{\mu})+\boldsymbol{\mu}(\boldsymbol{\delta}_e)$ and the covariance $\hat{\boldsymbol{\Sigma}}_e = \boldsymbol{\kappa}(\boldsymbol{\delta}_e, \boldsymbol{\delta}_e) - \mathbf{k}_e^T(\mathbf{K}+\boldsymbol{\Sigma})^{-1}\mathbf{k}_e$. Note that in general, the choice of the mean and kernel functions is important and

specific to the type of robot in use. In the present case, we use the linear mean function

$$\boldsymbol{\mu}(\mathbf{x}) = \mathbf{C}\mathbf{x} \tag{6}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the vector of wheel ticks recorded in a time interval and $\mathbf{C} \in \mathbb{R}^{3 \times m}$ is the associated hyper-parameter of the mean function. Recall that m represents total number of wheels equipped with wheel encoders. Intuitively, the implication of this choice of linear mean function is that the relative position of the robot varies linearly with the wheel ticks recorded in the corresponding time interval. Such a relationship generally holds for arbitrary drive configurations if the time interval is sufficiently small. A widely used kernel function is the radial basis function as follows

$$[\boldsymbol{\kappa}_{rbf}(\mathbf{x}, \mathbf{x}')]_{i,i'} = \sigma_{i,i'}^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{B}_{i,i'}^{-1}(\mathbf{x} - \mathbf{x}')\right) \tag{7}$$

where $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^m$ are the data inputs with hyper-parameters $\Xi = [\sigma_{i,i'}, \mathbf{B}_{i,i'}]$, here $i, i' = 1, 2, 3$. It will be shown in section IV-C that for the two-wheel differential drive robot in use here, the squared exponential kernel (7) with the linear mean function yielded better results than others. On the other hand for four-wheel Mecanum drive in use here, the inner product kernel, which amounts to a linear transformation of the feature space,

$$[\boldsymbol{\kappa}_{lin}(\mathbf{x}, \mathbf{x}')]_{i,i'} = \langle \mathbf{x}, \mathbf{x}' \rangle \tag{8}$$

performed better. We remark here that for our experiments we have assumed $\boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}')$ is a diagonal matrix with diagonal entries $\{[\boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}')]_{i,i}\}$. In general, the choice of the mean and kernel functions and that of the associated hyper-parameters is made a priori. For our experiments we infer the hyper-parameters by optimizing the corresponding log marginal likelihood. However, they may also be determined during the calibration phase via cross-validation.

Algorithm 1 CGP algorithm to learn the motion model of the sensor

---
1: Collect measurements from sensors.
2: **Training Phase :**
3: Run sensor displacement algorithm for each selected interval, to get the estimates $\{\hat{\mathbf{s}}_{jk}\}$ with the corresponding wheel ticks $\boldsymbol{\delta}_{jk}$ and stack them.
4: Now pre-compute the following quantities :
5: $\quad (\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}(\hat{\mathbf{s}} - \bar{\boldsymbol{\mu}})$ and $(\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}$
6: **Testing Phase :**
7: For every test input $\boldsymbol{\delta}_e$, evaluate the following,
8: $\quad \hat{\boldsymbol{\mu}}_e = \boldsymbol{k}_e(\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}(\hat{\mathbf{s}} - \bar{\boldsymbol{\mu}}) + \boldsymbol{\mu}(\boldsymbol{\delta}_e)$
9: $\quad \hat{\boldsymbol{\Sigma}}_e = \boldsymbol{\kappa}(\boldsymbol{\delta}_e, \boldsymbol{\delta}_e) - \mathbf{k}_e^T(\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}\mathbf{k}_e$
10: Report $\hat{\mathbf{s}}_e$, where $p(\hat{\mathbf{s}}_e) = \mathcal{N}(\hat{\mathbf{s}}_e|\hat{\boldsymbol{\mu}}_e, \hat{\boldsymbol{\Sigma}}_e)$

B. Approximate linear motion model

As an alternative to the general and flexible CGP approach that is applicable to any robot, we also put forth a computationally simple approach that relies on a linear approximation of $\boldsymbol{f}$. Specifically, if Δt_{jk} is sufficiently small,

(a) x   (b) y   (c) θ
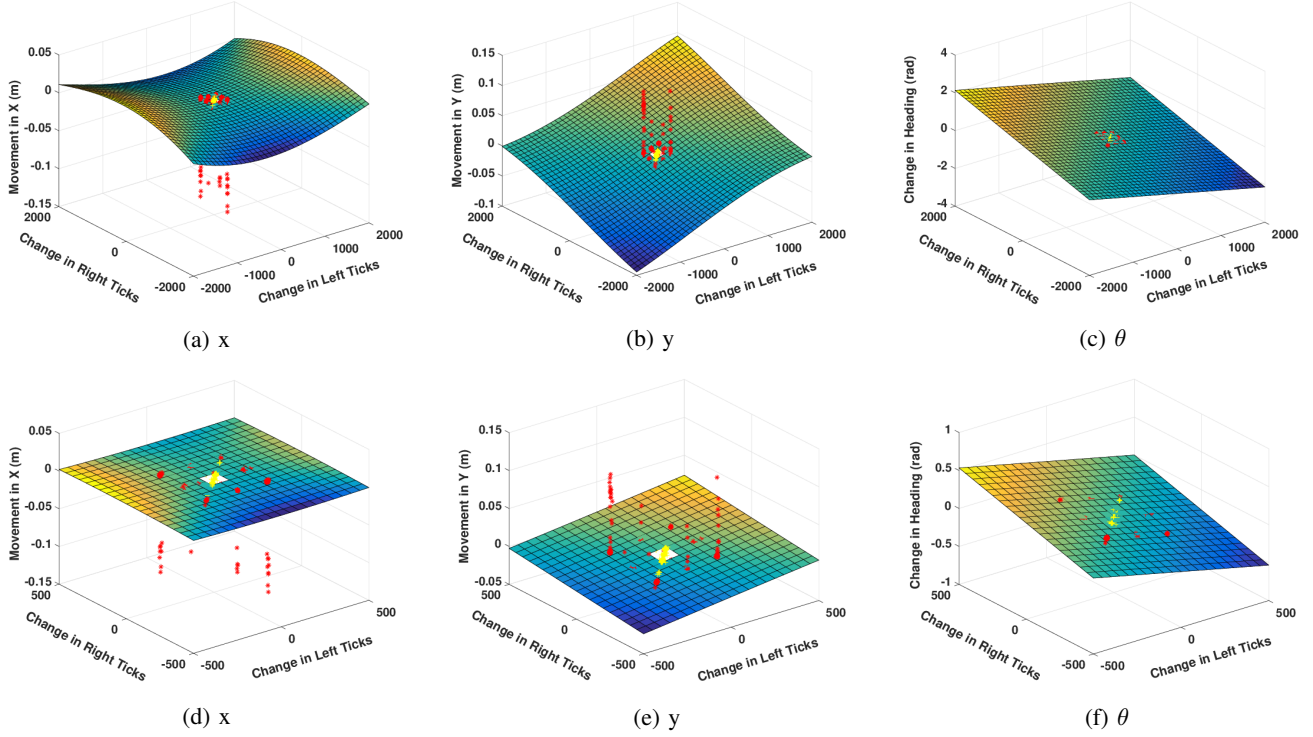
(d) x   (e) y   (f) θ

Fig. 3: (a),(b),(c) illustrates the movement of the sensor frame in x, y, θ, respectively w.r.t change in left and right wheel ticks of a two wheel differential drive robot (i.e., $\boldsymbol{f}(\bullet) = \boldsymbol{g}(\bullet ; \hat{\boldsymbol{p}})$), overlaid with the corresponding sensor displacement measurements generated using raw data published at [18] for a particular configuration. Note $\hat{\boldsymbol{p}}$ denotes parameter estimates found using CMLE [10]. Also in [18], since data from each configuration is divided into three subsets, we consider any two of them as training data and rest as test data. Points that are displayed in red and yellow color denote training and testing samples generated at selected scan instants respectively. Note: red points that are away from the 3D surface are outliers. (d)-(f) represent the truncated and enlarged versions of the same plots to expose the linearity. (Figure is best viewed in color)

so are elements of $\boldsymbol{\delta}_{jk}$. Therefore, it follows from the first order Taylor's series expansion, that $\boldsymbol{f}$ is approximately linear. This assertion if further verified empirically for the two-wheel differential drive. As evident from Fig. 3, for Δt_{jk} sufficiently small, the elements of $\boldsymbol{\delta}_{jk}$ are concentrated around zero and the surface fitting them is indeed approximately linear. Motivated by the observation in Fig. 3, we let $\boldsymbol{f}(\boldsymbol{\delta}_{jk}) = \mathbf{W}\boldsymbol{\delta}_{jk}$, where $\mathbf{W} \in \mathbb{R}^{3 \times m}$ is the unknown weight matrix. The following robust linear regression problem can subsequently be solved to yield the weights:

$$\widehat{\mathbf{W}} = \arg \min_{\mathbf{W}} \sum_{(j,k) \in \mathcal{E}} \sum_{i \in \{x,y,\theta\}} \rho_c \left(\frac{\hat{\mathbf{s}}_{jk}^i - [\mathbf{W}\boldsymbol{\delta}_{jk}]_i}{\sigma_{jk}^i} \right) \quad (9)$$

where ρ_c is the Huber loss function [19]. Here, (9) is a convex optimization problem and can be solved efficiently with complexity $\mathcal{O}(n^3)$. It is remarked that the entries of $\mathbf{W}$ do not have any physical significance and cannot generally be related to the intrinsic or extrinsic robot parameters, especially after wheel deformation. Note that while making predictions the complexity of the linear model is $\mathcal{O}(m)$ where as for CGP it is $\mathcal{O}(n^2)$.

IV. Experimental Evaluations

This section details the experiments carried out to test the proposed CGP algorithm. We begin with the performance metrics used for validating the accuracy of the estimated model followed by details regarding the experimental setup and results.

A. Performance Metrics

In the absence of wheel slippages, it is remarked that the accuracy of estimated model is quantified by the closeness of the robot/sensor trajectory estimate obtained from odometry to the ground truth trajectory. Since ground truth data was not available for the experiments, we instead used a SLAM algorithm to localize the sensor and build a map of the environment. While SLAM output would itself be not as accurate as compared to the ground truth, of which some of them [20] do not require odometry measurements and consequently serves as a benchmark for all calibration algorithms. Specifically, the *google cartographer* algorithm, which leverages a robust scan to sub-map joining routine, is used for generating the trajectory and the map [20] of the environments. It is remarked that in the absence of extrinsic calibration parameters, SLAM outputs only the sensor trajectory (and not the robot trajectory), which is subsequently used for comparisons.

Various sensor trajectory estimates are compared on the basis of Relative Pose Error (RPE) and the Absolute Trajectory Error (ATE) motivated from [21]. The RPE measures the local accuracy of the trajectory, and is indicative of the
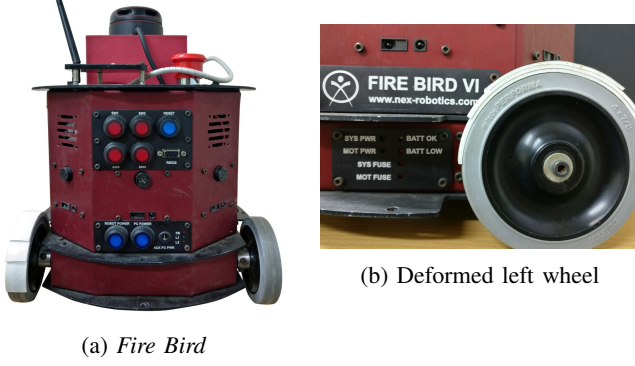
(a) *Fire Bird*

(b) Deformed left wheel

Fig. 4: *Fire Bird VI* robot used for experimental evaluations.

TABLE II: List of experimental configurations with labels and locations

Robot	Configuration	Training Data	Test Data
FireBird VI	**F1**	WSN Lab	Tomography Lab
Turtlebot3	**T1**	ACES Library	ACES Library
	T2		

drift in the estimated trajectory as compared to the ground truth. At any time $t_k \in \mathcal{T}$, let the odometry and SLAM pose estimates be denoted by $\hat{\boldsymbol{x}}_k$ and $\boldsymbol{x}_k$, respectively. Then, relative pose change between times t_k and t_{k+1} estimated via odometry and SLAM are given by $\ominus \, \hat{\boldsymbol{x}}_k \oplus \hat{\boldsymbol{x}}_{k+1}$ and $\ominus \, \boldsymbol{x}_k \oplus \boldsymbol{x}_{k+1}$, respectively. Defining $\mathbf{e}_k^r := \ominus \, (\ominus \, \hat{\boldsymbol{x}}_k \oplus \hat{\boldsymbol{x}}_{k+1}) \oplus (\ominus \, \boldsymbol{x}_k \oplus \boldsymbol{x}_{k+1})$, the RPE is defined as the root mean square of the translational components of $\{\mathbf{e}_k^r\}_{k=1}^{n-1}$, i.e.,

$$\text{RPE} := \left(\frac{1}{n-1} \sum_{k=1}^{n-1} \|\text{trans}(\mathbf{e}_k^r)\|^2 \right)^{1/2} \quad (10)$$

where $\text{trans}(\mathbf{e}_k)$ refers to the translational components of $\mathbf{e}_k$. In contrast, the ATE measures the global (in)consistency of the estimated trajectory and is indicative of the absolute distance between the poses estimated by odometry and SLAM at any time t_k. Defining the absolute pose error at time t_k as $\mathbf{e}_k^a := \ominus \, \hat{\boldsymbol{x}}_k \oplus \boldsymbol{x}_k$, the ATE is evaluated as the root mean square of the pose errors for all times $t_k \in \mathcal{T}$, i.e.,

$$\text{ATE} := \left(\frac{1}{n} \sum_{k=1}^{n} \|\text{trans}(\mathbf{e}_k^a)\|^2 \right)^{1/2} . \quad (11)$$

Next, we detail the experimental setup used to test model estimates from different forms of Gaussian process (GP).

TABLE III: ATE and RPE for Configurations **F1**

GP Estimate		Configuratin F1	
Mean fn	**Kernel fn**	**ATE (m)**	**RPE (mm)**
Zero	RBF	6.273	9.634
Linear	RBF	**0.592**	**9.367**
Zero	Linear	0.687	9.367
Zero	RBF + Linear	0.716	9.34
Linear	RBF + Linear	0.732	9.343
Linear Model		**0.687**	9.367
CMLE [10]		1.546	9.361

TABLE IV: ATE and RPE for Configurations **T1** and **T2**

GP Estimate		Configuration T2		Configuration T1	
Mean fn	**Kernel fn**	**ATE (m)**	**RPE (mm)**	**ATE (m)**	**RPE (mm)**
Zero	RBF	2.49	5.21	6.523	5.466
Linear	RBF	0.161	8.324	5.006	5.573
Zero	Linear	**0.068**	**5.108**	**0.87**	**5.457**
Zero	RBF + Linear	3.798	5.121	0.87	5.455
Linear	RBF + Linear	1.193	5.49	1.849	5.519
Linear Model		**0.068**	**5.108**	0.869	**5.458**
Regular Model		4.24	5.112	3.647	5.517

B. Experimental Setup

1) Robots: We have used a two-wheel differential drive *FireBird VI* robot (see Fig. 4) having a particular set of intrinsic parameters [22] and a four-wheel mecanum drive *Turtlebot3* robot (see Fig. 1). The *Fire Bird VI* is primarily a research robot with diameter 280 mm, weight of 12 kilograms, and maximum translational velocity of 1.28 m/s. All *FireBird* encoders publish data at the rate of 10 Hz with a resolution of 3840 ticks per revolution. Similarly *Turtlebot3 mecanum* is also a research robot from the Robotis group with all wheels diameter of 60 mm. It weighs 1.8 kilograms and maximum translational velocity is 0.26m/s. The dynamixels used publish data at 10 Hz with an approximate resolution of 4096 ticks per revolution. For the purposes of the experiment, we made use of an on board computer with i5 processor, 8GB RAM, running ROS kinetic for processing the data from lidar and wheel encoders, performing SLAM for validation, and running the calibration algorithms.

2) Lidar Sensor: RPLidar A2 is a low cost $360°$, $2D$ laser scanner with a detection range of 6 meters, a distance resolution less than 0.5 m and an adjustable operating frequency of 5 to 15 Hz. This scanner was mounted on the both the robots with the frequency of 10 Hz resulting in an angular resolution of $0.9°$.

3) Scan Matching: We used point-to-line ICP (PLICP) variant [23] in order to estimate the sensor displacements $\hat{\mathbf{s}}_{jk}$. It is remarked that all ICP-like methods also output the corresponding covariance value in closed-form [24] that can be used by the CGP algorithm.

(a) Tomography Lab, Configuratoin **F1**   (b) ACES Library, Configuration **T1**   (c) ACES Library, Configuration **T2**
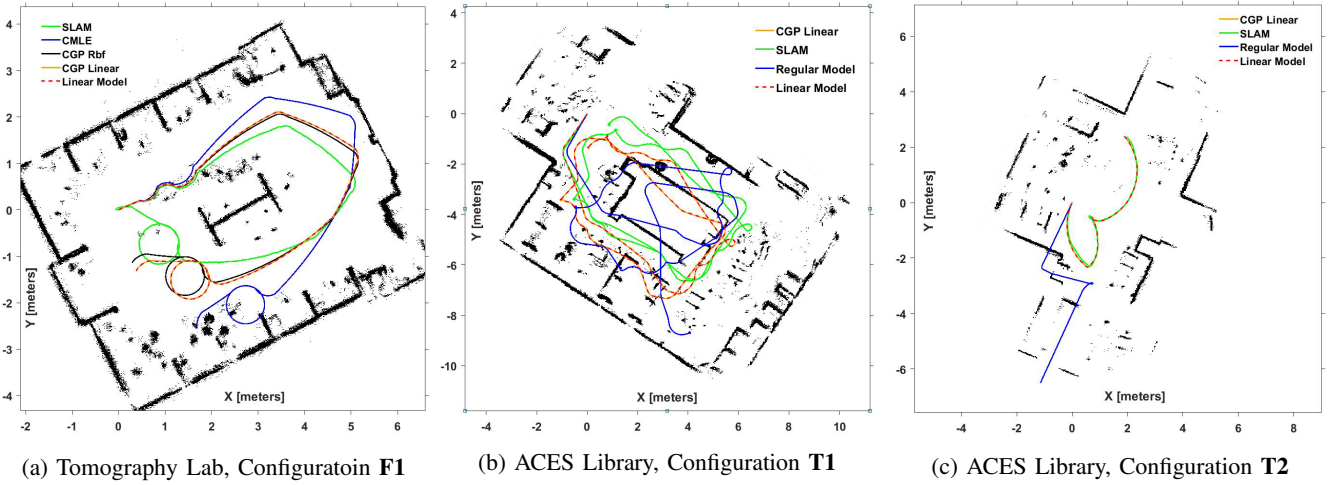
Fig. 5: Trajectory comparison against SLAM for different robot configurations. (a) is the test environment for the configuration **F1** where as (b) and (c) are for configurations **T1** and **T2** respectively.

4) Data Processing: For the purposes of the experiments, we ensure that scans are collected at times spaced T seconds apart. The choice of T is not trivial. For instance, choosing a small T often makes the algorithm too sensitive to un-modeled effects arising due to synchronization of sensors, robot's dynamics. It is lucrative to choose far scan pairs as more information is capured about the parameters however both the scan matching output as well as the motion model become inaccurate when T is large. For the experiments, we chose the largest value of T that yielded a reliable scan matching output in the form of sensor motion, resulting in $T = 0.6$ second for *Turtlebot3 mecanum* and $T = 0.3$ seconds for the *Firebird VI* robot. These values are chosen based on maximum wheel speeds such that slippages are minimized during experimentation. Note that since the odometry readings are acquired at a rate, higher than the scans, temporally closest odometry reading is associated to a given scan. With the chosen T the robots would move a maximum displacement of 15cm in x and y and $8°$ in yaw, under such conditions PLICP achieves 99.51 % accuracy [23].

5) Deformed Robot Configurations: In order to demonstrate the non-availability of the robot model, one of the wheels of the *Firebird VI* robot is deformed with a thick tape (see Fig. 4(b)), this configuration is referred as **F1**. Care was taken to ensure that the deformation was not too large, so as to avoid wobbling of the robot and the scan plane of the Lidar. In the case of *Turtlebot3* robot two different configurations (**T1** and **T2**) are constructed (see Fig. 1), by changing the position of the wheels from the regular config-uration. We will see further that the amount of deformation in tilted wheel configuration **T2** (as in Fig. 1(b)) is more as opposed to unaligned wheels configuration **T1** (as in Fig. 1(a)). Next, experiments comprising of training and testing phases, are carried out using these deformed robots for all the specified configurations (see Table II). While training data is used to learn the motion model of the robot/sensor, the test data is used to evaluate the accuracy of the learned model. It is remarked that the collected test data involves

short and long trajectories with varied robot motions. Each experiment is labeled for reference, with details provided as shown in Table II. For example, configuration **T1** refers to the experiment done using *Turtlebot3* robot, where both training and test data are collected in ACES library.

C. Experimental Results

We first perform offline calibration of *FireBird VI* robot with configuration **F1** using the proposed CGP algorithm along with the model based CMLE [10] algorithm. Note that in the case of CGP algorithm various kernel and mean func-tions are trained to determine which of them captures the sen-sor motion model accurately. Note that we have also trained on composite kernel functions like $\kappa_{rbf} + \kappa_{lin}(\mathbf{x}, \mathbf{x}')$. After the model is learned, predictions are made on the test data. The predicted trajectories are then compared with SLAM trajectory as reference. Error metrics for these trajectories are generated and displayed in Table III. It is observed that CGP with squared exponential kernel function with linear mean function outperforms other trained models, also CMLE. We remark here that although CMLE predicts the radius of the left wheel to be slightly more than that of the right wheel, the predictions are worse due to non applicability of the parametric model as the wheel looses its notion of circularity. Observe that CGP with linear kernel function is comparable to the best case. The predicted trajectories generated for CMLE [10], CGP with linear kernel and SLAM [20] are displayed in Fig. 5(a). It is evident that the proposed CGP with linear kernel predicts test trajectory close to SLAM.

Similar procedure is carried out in the case of Turtlebot3 robot with configurations **T1** and **T2**. Note that since the analysis of CMLE [10] is restricted to two wheel differen-tial drive robots, we use parametric motion model of four wheel mecanum drive robot [16] with manufacturer specified parameters for robot intrinsics and nominal hand measured parameters for lidar extrinsics to perform predictions on test data. Table IV displays error metrics evaluated for parametric and various non-parametric models. Observe that the proposed CGP algorithm with linear kernel function

outperforms other learned models. The corresponding test trajectories for configurations **T1** and **T2** are displayed in Fig. 5(b) and Fig. 5(c) respectively.

Interestingly it can be observed from Table.III and Table IV that the linear model approximation is sufficient to explain the motion model with the set deformities in all configurations. Here we notice that learning a linear approximation of f is sufficient to accurately predict robot odometry, this is in lines with our discussion in Sec. III-B.

V. CONCLUSION

We develop a novel odometry and sensor calibration framework applicable to wheeled mobile robots operating in planar environments. The key idea is to utilize the ego-motion estimates from the exteroceptive sensor to estimate the motion model of the sensor/robot. The proposed framework is general as it applies to robots whose motion model is not known. We advocate a non-parametric Gaussian process regression-based approach that directly learns the relationship between the wheel odometry and the sensor motion. The method does not require ground-truth measurements from an external setup, and henceforth the calibration routine can be carried out without interrupting the robot operation. A computationally efficient method that relies on a linear approximation of the sensor motion model is shown to perform on par with the proposed calibration via Gaussian process (CGP) algorithm. Experiments are performed on robots with un-modelled deformations and is shown to outperform existing parametric approaches. Moreover, all the MATLAB codes are made available online[1]. The method being general is applied to wheeled robots operating in planar environments but does not make any assumptions regarding the same. As part of the future work, it would be interesting to test the performance in non-planar settings.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *Proc. IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, June 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *Proc. IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, Sept 2006.

[3] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: a multi-tier camera sensor network," *Proc. of the 13th annual ACM International Conference on Multimedia*, pp. 229–238, 2005.

[4] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[5] J. Hidalgo-Carrió, D. Hennes, J. Schwendner, and F. Kirchner, "Gaussian process estimation of odometry errors for localization and mapping," *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5696–5701, 2017.

[6] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." *Robotics: Science and Systems*, vol. 2, 2014.

[7] Z. Fang, S. Yang, S. Jain, G. Dubey, S. Roth, S. Maeta, S. Nuske, Y. Zhang, and S. Scherer, "Robust autonomous flight in constrained and visually degraded shipboard environments," *Proc. Journal of Field Robotics*, vol. 34, no. 1, pp. 25–52, 2017.

[8] A. Kelly, "Fast and easy systematic and stochastic odometry calibration," *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 4, pp. 3188–3194 vol.4, Sept 2004.

[9] K. Lee and W. Chung, "Calibration of kinematic parameters of a car-like mobile robot to improve odometry accuracy," *Proc. IEEE International Conference on Robotics and Automation*, pp. 2546–2551, 2008.

[10] A. Censi, A. Franchi, L. Marchionni, and G. Oriolo, "Simultaneous calibration of odometry and sensor parameters for mobile robots," *Proc. IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 475–492, April 2013.

[11] F. Kallasi, D. L. Rizzini, F. Oleari, M. Magnani, and S. Caselli, "A novel calibration method for industrial agvs," *Robotics and Autonomous Systems*, vol. 94, pp. 75–88, 2017.

[12] D. A. Cucci and M. Matteucci, "A flexible framework for mobile robot pose estimation and multi-sensor self-calibration." *ICINCO (2)*, pp. 361–368, 2013.

[13] R. Kümmerle, G. Grisetti, and W. Burgard, "Simultaneous calibration, localization, and mapping," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3716–3721, 2011.

[14] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2007, pp. 742–747.

[15] H. Schulz, L. Ott, J. Sturm, and W. Burgard, "Learning kinematics from direct self-observation using nearest-neighbor methods," *Advances in Robotics Research*, pp. 11–20, 2009.

[16] H. Taheri, B. Qiao, and N. Ghaeminezhad, "Kinematic model of a four mecanum wheeled mobile robot," *International journal of computer applications*, vol. 113, no. 3, pp. 6–9, 2015.

[17] H. Tang and Y. Liu, "A fully automatic calibration algorithm for a camera odometry system," *Proc. IEEE Sensors Journal*, vol. 17, no. 13, pp. 4208–4216, 2017.

[18] A. Censi. Supplemental material for simultaneous calibration of odometry and sensor parameters for mobile robots. [Online]. Available: https://github.com/AndreaCensi/calibration

[19] P. J. Huber, *Robust statistical procedures*. SIAM, 1996.

[20] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[21] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 573–580, 2012.

[22] Find about fire bird vi robotic research platform. [Online]. Available: http://www.nex-robotics.com/products/fire-bird-vi-robot/fire-bird-vi-robotic-research-platform.html

[23] A. Censi, "An ICP variant using a point-to-line metric," *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 19–25, May 2008.

[24] ——, "An accurate closed-form estimate of icp's covariance," *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3167–3172, 2007.

[1] http://www.tinyurl.com/GPcalibration