Extrinsic Calibration of an Eye-In-Hand 2D LiDAR Sensor in Unstructured Environments Using ICP

Arne Peters, Adam Schmidt, and Alois C. Knoll

***Abstract*—We propose a calibration method for the six degrees of freedom (DOF) extrinsic pose of a 2D laser rangefinder mounted to a robot arm. Our goal is to design a system that allows on-site re-calibration without requiring any kind of special environment or calibration objects. By moving the sensor we generate 3D scans of the surrounding area on which we run a iterative closest point (ICP) variant to estimate the missing part of the kinematic chain. With this setup we can simply scale the density and format of our 3D scan by adjusting the robot speed and trajectory, allowing us to exploit the power of a high resolution 3D scanner for a variety of tasks such as mapping, object recognition and grasp planning. Our evaluation, performed on synthetic datasets as well as from real-data shows that the presented approach provides good results both in terms of convergence on crude initial parameters as well as in the precision of the final estimate.**

***Index Terms*—Calibration and identification, computer vision for other robotic applications, range sensing, sensor fusion, 3D reconstruction.**

I. Introduction

EYE-IN-HAND sensors offer a great potential to mobile robotics as they allow capturing of data that is not visible to their static counterparts. Just imagine a service or logistics robot detecting the contents of a shelf by simply scanning it from top to bottom.

To make use of such scan data the relative pose of the sensor to the robots end-effector needs to be exactly known. While calibration is often considered a once-in-a-lifetime task there are situations in which it is essential to be able to re-calibrate a system on-site. If a system becomes uncalibrated after a repair, a collision with its environment or other mechanical stress as i.e. from transportation, the results of its processing of sensor data can become worse or even useless. The consequences of such a situation can reach from simple loss cost of time or money over an abortion of mission in case the re-calibration cannot be performed on-site, up to a complete loss of the robotic system if it gets stuck in a hostile surrounding.

Thus our goal was to design a calibration approach that does not require any knowledge nor influence on the robots environment. In detail our contributions are:

1) We present a simple approach to filter noise like reflections or laser shadows from distorted scan data
2) We show an ICP variant to calibration the mounting pose of a 2D lidar sensor on a robot arm – which is to our best knowledge the first solution to this problem capable of estimating the full 6 DOF pose of such an eye-in-hand sensor without using an external calibration object
3) We demonstrate that our approach gives good results, both in terms of convergence and precision by testing it on multiple synthetic datasets as well as on real-data

II. Related Work

The problem of calibrating eye-in-hand laser sensors is not new. Especially with the size and price reduction of such devices over the last decade many approaches for calibrating such systems have been demonstrated. However almost all of them rely on using known calibration targets and/or the known pose of certain objects.

Wagner *et al.* [1] systematically move a triangulation based 2D laser scanner in x and y direction to find the tip of pin placed next to the robot. In [2] the surface of a sphere is detected in depth data and optimized for its center over multiple measurements taken from different robot poses. In the works of Antone and Friedman [3], a special calibration object is designed which allows the estimation of all calibration parameters from a single scan. Andersen, Andersen and Ravn [4] estimate the pose of a statically mounted laser rangefinder by moving a board with a cut-out triangle by a robot. The approach does not offer very high precision, but works in the very limited space between the sensor and a conveyor belt.

The PR2 calibration package [5] estimated the extrinsic position of a laser rangefinder installed on a tilting joint in the robots torso by detecting a checkboard held by the robotic arm in the lasers intensity data. By detecting multiple intersection points on a known grid the approach allows not only to estimate the transformation between the laser and the joint it is mounted to, but also to optimize the whole kinematic chain of the robot.

Scaramuzza, Harati and Siegwart [6] avoided the use of calibration targets by working with manually annotated

Manuscript received September 10, 2019; accepted December 31, 2019. Date of publication January 13, 2020; date of current version January 29, 2020. This letter was recommended for publication by Associate Editor H. Zhang and Editor D. Song upon evaluation of the reviewers' comments. This letter is part of projects that have received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement Nos. 680734 and 870133. *(Corresponding author: Arne Peters.)*

A. Peters and A. C. Knoll are with the Chair of Robotics, Artificial Intelligence and Real-time Systems, Technical University of Munich, 85748 Garching, Germany (e-mail: arne.peters@tum.de; knoll@in.tum.de).

A. Schmidt was with the Technical University of Munich, 85748 Garching, Germany, and is now with the Netherlands Organization for Applied Scientific Research (TNO), NL-5656 AE Eindhoven, The Netherlands (e-mail: adam.schmidt@tno.nl).

Digital Object Identifier 10.1109/LRA.2020.2965878

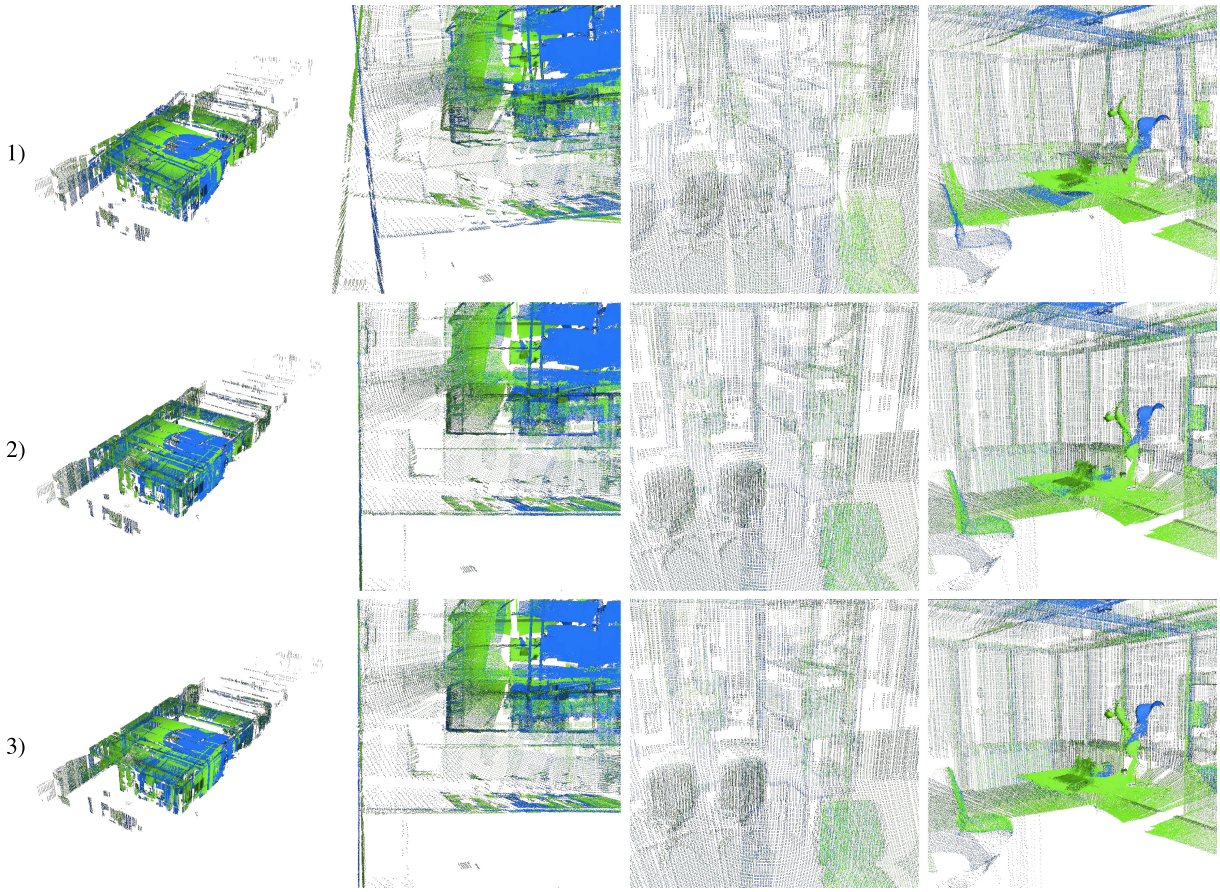


Fig. 1. Details of a 3D scan from of our laboratory. 1) Shows a crude initial guess for the extrinsic parameters, 2) is rendered using the results from our calibration initialized with the guess from 1. The 3D scan in 3) is based on parameters from the CAD data of our 3D printed sensor mount.

correspondences to calibrate the extrinsic transformation between camera and a 3D laser scanner.

A full self-calibration similar to ours has been shown by [7]: Sheehan, Harrison and Newman calibrate a system of three Sick Lidar sensors revolving around a single axis by optimizing the "crispness of corners" (squared Renyi entropy) in the resulting point cloud of the fused measurements. One drawback of this idea is the computational effort required to compute the residual function, as computing the crispness requires a comparison between every single measurement and all other points of the dataset. Moreover, due to this limited motion pattern of the sensors they were only able to estimate 4 DOF: As the sensors only move in a plane it is impossible to calculate their position along the rotation axis as well as the offset from the rotation axis' zero angle. [8] applied this approach to a laser following a swiveling motion pattern [9]. On top they show that working with a down-scaled copy of the original dataset gives the same results as working with the full scan but requires less computing time.

Also the use of the Iterative-Closest-Point (ICP) algorithm [10], [11] for calibration purposes has been demonstrated before. Many works use ICP to estimate the transformations between multiple sensors by registering scan data [12]–[14]. [15] calibrates the pose of multiple lidar sensors mounted on an excavator by matching the scan data against the CAD data of the boom and shovel. Similar to us Alismail, Baker and Browning [16], [17] use a point-to-plane ICP to estimate the transformation between a rotating lidar sensor and it's rotation axis based on redundancies in the scan data. However, just like Sheehan they worked with only one rotation axis. To get the full 6 DOF transformation to an additional camera they introduce a planar calibration target, that can also be detected in the camera images.

III. Approach

In our set-up we are working with a Hokuyo UTM-30LX laser range finder attached to a KUKA LBR iiwa 14 robot arm as shown in figure 2. This configuration enables us to use a 2D range finder for 3D mapping of the environment with the flexibility of an eye-in-hand depth sensor for object detection and grasping.

We assume that the robot arm as well as the lasers intrinsic parameters are well calibrated. What is still missing is the transformation $\boldsymbol{T}^{\mathcal{L}\to\mathcal{E}}$ between between the frames of the laser rangefinder $\mathcal{L}$ and the robots end-effector $\mathcal{E}$. $\boldsymbol{T}^{\mathcal{L}\to\mathcal{E}}$ is defined by 6 DOF: the three translation parameters $\boldsymbol{t} = (x, y, z)^{\mathrm{T}}$ and three rotation angles $\boldsymbol{r} = (\alpha, \beta, \gamma)^{\mathrm{T}}$ around roll, pitch and yaw,

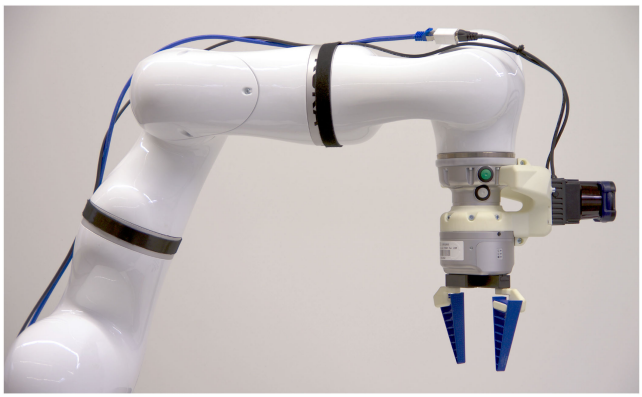


Fig. 2. KUKA LBR iiwa robot arm with attached Hokuyo UTM-30LX lidar sensor.

so that

$$\boldsymbol{T}^{\mathcal{L}\to\mathcal{E}} = \begin{bmatrix} \mathrm{ro}\left({\boldsymbol{e}_z}^{\mathcal{L}}, \gamma\right) \cdot \mathrm{ro}\left({\boldsymbol{e}_y}^{\mathcal{L}}, \beta\right) \cdot \mathrm{ro}\left({\boldsymbol{e}_x}^{\mathcal{L}}, \alpha\right) & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \tag{1}$$

where ${\boldsymbol{e}_x}^{\mathcal{L}}$, ${\boldsymbol{e}_x}^{\mathcal{L}}$ and ${\boldsymbol{e}_x}^{\mathcal{L}}$ are the basis vectors of $\mathcal{L}$ and $\mathrm{ro}(\boldsymbol{a}, \psi)$ gives the angle-axis rotation matrix of ψ around a normalized axis $\boldsymbol{a}$:

$$\mathrm{ro}\left(\boldsymbol{a}, \psi\right) \mapsto \left(\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\right) \tag{2}$$

with

$$\boldsymbol{w}_1 \begin{pmatrix} \cos\psi + a_x^2\left(1-\cos\psi\right) \\ a_y a_x\left(1-\cos\psi\right) + a_z \sin\psi \\ a_z a_x\left(1-\cos\psi\right) - a_y \sin\psi \end{pmatrix}, \tag{3}$$

$$\boldsymbol{w}_2 \begin{pmatrix} a_x a_y\left(1-\cos\psi\right) - a_z \sin\psi \\ \cos\psi + a_y^2\left(1-\cos\psi\right) \\ a_z a_y\left(1-\cos\psi\right) + a_x \sin\psi \end{pmatrix} \tag{4}$$

and

$$\boldsymbol{w}_3 \begin{pmatrix} a_x a_z\left(1-\cos\psi\right) + a_y \sin\psi \\ a_y a_z\left(1-\cos\psi\right) - a_x \sin\psi \\ \cos\psi + a_z^2\left(1-\cos\psi\right) \end{pmatrix}. \tag{5}$$

Our idea is to take two 3D scans of the robots environment from different robot configurations, each by revolving the last joint of the robot from $-90°$ to $90°$. As the laser has a viewing angle of $270°$ a $180°$ rotation is sufficient to capture a full scan of the surroundings. We then transform the scans to the coordinate frame of the robot's base $\mathcal{B}$ (see figure 3) and use the ICP algorithm to find the optimal values for $\boldsymbol{t}$ and $\boldsymbol{r}$ to complete our kinematic chain.

A. Initialization and Preprocessing

For each measurement at time t we know the measured depth d_t as well as the orientation θ_t around the lasers scanning axis

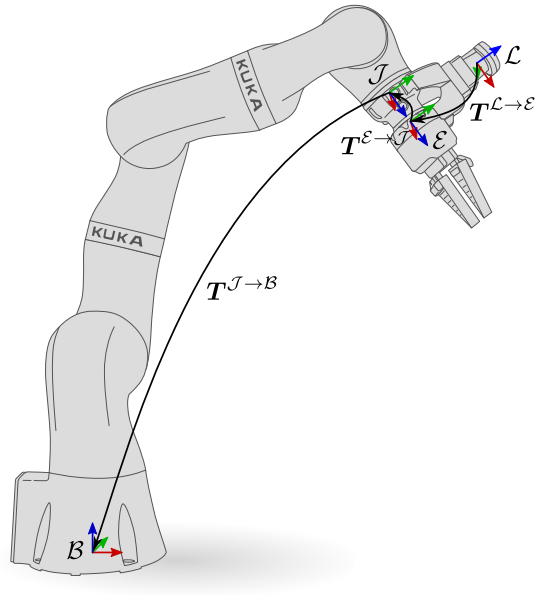


Fig. 3. Coordinate systems of our robot. Note that we compute the rotation of the wrist joint separately so its orientation is neither included in $\boldsymbol{T}^{\mathcal{E}\to\mathcal{J}}$ nor in $\boldsymbol{T}^{\mathcal{J}\to\mathcal{B}}$.

${\boldsymbol{e}_z}^{\mathcal{L}}$ allowing us to project the measurement to a point in $\mathcal{L}$ by

$$\boldsymbol{p}_t^{\mathcal{L}} = \begin{pmatrix} d_t \cos\left(\theta_t\right) \\ d_t \sin\left(\theta_t\right) \\ 0 \end{pmatrix}. \tag{6}$$

To fuse our measurements to a 3D point cloud we transform all $\boldsymbol{p}_t^{\mathcal{L}}$ to the frame of the last robot joint $\mathcal{J}$ via $\mathcal{E}$:

$$\begin{pmatrix} \boldsymbol{p}_t^{\mathcal{J}} \\ 1 \end{pmatrix} = \begin{bmatrix} \mathrm{ro}\left(\boldsymbol{j}, \varphi_t\right) & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{T}^{\mathcal{E}\to\mathcal{J}} \boldsymbol{T}^{\mathcal{L}\to\mathcal{E}} \begin{pmatrix} \boldsymbol{p}_t^{\mathcal{L}} \\ 1 \end{pmatrix} \tag{7}$$

where $\boldsymbol{T}^{\mathcal{E}\to\mathcal{J}}$ is the static offset between the end effector and φ_t the angle of the joint axis $\boldsymbol{j}$ (in our case ${\boldsymbol{e}_z}^{\mathcal{J}}$). Unfortunately, we do not get synchronized pose updates and depth measures so we do not have a direct reading of the arms joint positions for every t. In our setup pose updates are received at a rate of 100 Hz while the laser captures 40 scanlines, each with 1,080 depth measurements per second. However we are revolving with a constant speed, so that we can solve this problem by linear interpolation of the closest known angles. As the angles do not change in the further process we save the interpolated values in a vector $\boldsymbol{\phi}$.

As the environment has been recorded by spinning around ${\boldsymbol{e}_z}^{\mathcal{L}}$ and $\boldsymbol{j}$ we can arrange our points in the resulting point cloud $\boldsymbol{P}^{\mathcal{J}}$ along two indices along the axes:

$$\boldsymbol{P}^{\mathcal{J}} = \begin{bmatrix} \boldsymbol{p}_{0,0}^{\mathcal{J}} & \cdots & \boldsymbol{p}_{0,j_{\max}}^{\mathcal{J}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{p}_{i_{\max},0}^{\mathcal{J}} & \cdots & \boldsymbol{p}_{i_{\max},j_{\max}}^{\mathcal{J}} \end{bmatrix} \tag{8}$$

where i and j are the n^{th} measurement along θ and φ.

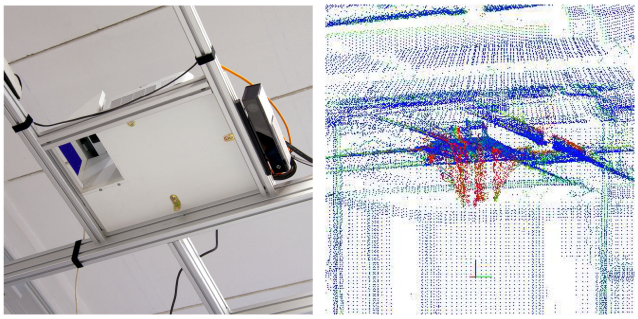


Fig. 4. Noisiness value for false measurements caused by a projector and the high gloss housing of a Kinect camera on the ceiling of our laboratory. Red is highest noise.

By doing so we can estimate the surface normal $\boldsymbol{n}_{i,j}$ for every $\boldsymbol{p}_{i,j}$ through the cross product of the differences with it's neighbors

$$\boldsymbol{n}_{i,j} = \mathrm{n}\left(\boldsymbol{p}_{i,j}\right) \mapsto \frac{\mathrm{c}\left(\boldsymbol{u}_{i,j}, \boldsymbol{r}_{i,j}\right) + \mathrm{c}\left(\boldsymbol{d}_{i,j}, \boldsymbol{l}_{i,j}\right)}{\left\|\mathrm{c}\left(\boldsymbol{u}_{i,j}, \boldsymbol{r}_{i,j}\right) + \mathrm{c}\left(\boldsymbol{d}_{i,j}, \boldsymbol{l}_{i,j}\right)\right\|} \tag{9}$$

with the vectors (up, right, down and left)

$$\begin{aligned} \boldsymbol{u}_{i,j} &= \boldsymbol{p}_{i-1,j} - \boldsymbol{p}_{i,j}, \\ \boldsymbol{r}_{i,j} &= \boldsymbol{p}_{i,j-1} - \boldsymbol{p}_{i,j}, \\ \boldsymbol{d}_{i,j} &= \boldsymbol{p}_{i+1,j} - \boldsymbol{p}_{i,j}, \\ \boldsymbol{l}_{i,j} &= \boldsymbol{p}_{i,j+1} - \boldsymbol{p}_{i,j} \end{aligned} \tag{10}$$

and

$$\mathrm{c}\left(\boldsymbol{a}, \boldsymbol{b}\right) = \frac{\boldsymbol{a} \times \boldsymbol{b}}{\left\|\boldsymbol{a} \times \boldsymbol{b}\right\|}. \tag{11}$$

As a next initialization step, we build up a validity mask $\boldsymbol{M}$ that indicates what points to exclude from the further process with $m_n = 0$ for points $\boldsymbol{p}_n$ with invalid depth values, too little intensity and/or missing neighbors for the normal estimation and $m_n = 1$ otherwise. We use a filter to mask points whose probability to be outliers, reflections or "laser shadows" (depth measurements that lie in the air between a foreground and a background object; see Figure 4) exceed a certain threshold. This "noisiness" of a point is estimated by

$$\mathrm{f}\left(\boldsymbol{p}_{i,j}, r\right) = \frac{\boldsymbol{s}_2 - \boldsymbol{s}_3}{\boldsymbol{s}_1 - \boldsymbol{s}_3} \tag{12}$$

with $\boldsymbol{s}$ containing the singular values of a row-wise matrix of the neighboring points of $\boldsymbol{p}_{i,j}$ in an r by r window (where r is an arbitrary parameter).

To improve the numerical stability of solution we calculate the scaling factor of $\boldsymbol{P}_1^{\mathcal{J}}$ based on our initial guess

$$s = \mathrm{size}\left(\boldsymbol{P}, \boldsymbol{M}\right) \mapsto \frac{\sum_{i=0}^{n} m_i \|\boldsymbol{p}_i\|}{\sum_{i=0}^{n} m_i} \tag{13}$$

and use it to create two scaled copies $\boldsymbol{Q}_1^L$ and $\boldsymbol{Q}_2^L$ of $\boldsymbol{P}_1^L$ and $\boldsymbol{P}_2^L$:

$$\boldsymbol{q}_i^{\mathcal{L}} = \frac{1}{s} \cdot \boldsymbol{q}_i^{\mathcal{L}} \forall \boldsymbol{q}_i^{\mathcal{L}} \in \boldsymbol{Q}^{\mathcal{L}} \tag{14}$$

so that the average distance of all points from the origin becomes 1. We can stick with our initial s for all ICP iterations as the calibration is meant to fine tune the laser pose which is usually not more than a few centimeters off, and thus has almost no impact on the overall dimensions of the point cloud. The same scaling also needs to be applied to the transformations $\boldsymbol{T}_1^{\mathcal{J}\to\mathcal{B}}$ and $\boldsymbol{T}_2^{\mathcal{J}\to\mathcal{B}}$:

$$\boldsymbol{T}_{i,\mathrm{scaled}}^{\mathcal{J}\to\mathcal{B}} = \mathrm{sc}\left(\boldsymbol{T}^{\mathcal{J}\to\mathcal{B}} i, \frac{1}{s}\right) \forall i \in \{1, 2\} \tag{15}$$

with

$$\mathrm{sc}\left(\boldsymbol{T}, f\right) \mapsto \begin{bmatrix} \boldsymbol{R} & f\boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \forall \boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix}. \tag{16}$$

As $\boldsymbol{T}^{\mathcal{E}\to\mathcal{J}}$ is just a constant offset to $\boldsymbol{T}^{\mathcal{L}\to\mathcal{E}}$, we can simplify the problem by looking for $\boldsymbol{T}^{\mathcal{L}\to\mathcal{J}}$ instead. Moreover, we convert the extrinsic parameters to $\mathbf{s} = {}^1\!/_s\mathbf{t}$ and $\boldsymbol{u} = (u_x, u_y, u_z)^{\mathrm{T}}$, where $\boldsymbol{u}$ defines a rotation axis vector and it's length indicates the rotation angle, so that

$$\mathrm{tf}\left(\boldsymbol{s}, \boldsymbol{u}\right) = \boldsymbol{T}_{\mathrm{scaled}}^{\mathcal{L}\to\mathcal{J}} = \mathrm{sc}\left(\boldsymbol{T}^{\mathcal{E}\to\mathcal{J}}, \frac{1}{s}\right) \boldsymbol{T}_{\mathrm{scaled}}^{\mathcal{L}\to\mathcal{E}} \tag{17}$$

with

$$\mathrm{tf}\left(\boldsymbol{s}, \boldsymbol{u}\right) \mapsto \begin{bmatrix} \mathrm{ro}\left(\frac{1}{\|\boldsymbol{u}\|}\boldsymbol{u}, \|\boldsymbol{u}\|\right) & \boldsymbol{s} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix}. \tag{18}$$

Representing the orientation in Rodriguez notation has shown to be numerically more stable than using Euler angles in our test.

B. Iterative Optimization

After all data has been initialized, we start an iterative optimization process which follows the common ICP procedure of searching for point matches and minimizing the sum of the error function over all matches by finding the best transformation between the two scans as shown in Algorithm 1.

We begin the optimization loop by transforming both scaled 3D scans $\boldsymbol{Q}_n^{\mathcal{L}} | n \in \{1, 2\}$ to $\mathcal{B}$

$$\begin{pmatrix} \boldsymbol{q}_i^{\mathcal{B}} \\ 1 \end{pmatrix} = \boldsymbol{T}^{\mathcal{J}\to\mathcal{B}} \mathrm{scaled} \begin{bmatrix} \mathrm{ro}\left(\boldsymbol{j}, \varphi_i\right) & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \mathrm{tf}\left(\boldsymbol{s}, \boldsymbol{u}\right) \begin{pmatrix} \boldsymbol{q}_i^{\mathcal{L}} 1 \end{pmatrix}. \tag{19}$$

After this we create a k-d tree [18] of all $\boldsymbol{q}_b^1 \in \boldsymbol{Q}_2^{\mathcal{B}} | m_b^2 = 1$ and use it to find pairs $\langle a, b \rangle$ of closest neighbors for every $\boldsymbol{q}_a^1 \in \boldsymbol{Q}_1^{\mathcal{B}} | m_a^1 = 1$. Similar to [19] two filters

$$\mathrm{f}_1(a, b) \mapsto \begin{cases} 1 & \left\|\boldsymbol{q}_a^1 - \boldsymbol{q}_b^2\right\| \leq t_1 \\ 0 & \text{else} \end{cases} \tag{20}$$

and

$$\mathrm{f}_2(a, b) \mapsto \begin{cases} 1 & n(\boldsymbol{p}_a^1) \cdot n(\boldsymbol{q}_b^2) \geq t_2 \\ 0 & \text{else} \end{cases} \tag{21}$$

are applied to mask matches between points that are too far away from each other or have normals pointing into different

Algorithm 1: ICP for Calibration.

Require: $\boldsymbol{P}_1^L$ and $\boldsymbol{P}_2^L$ along with $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$, $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$
Require: Initial guesses $\boldsymbol{t}_0$ and $\boldsymbol{r}_0$
Require: $\boldsymbol{T}^{\mathcal{E}\to\mathcal{J}}$, $\boldsymbol{T}_1^{\mathcal{J}\to\mathcal{B}}$ and $\boldsymbol{T}_2^{\mathcal{J}\to\mathcal{B}}$
Require: Stop threshold c
  Compute $\boldsymbol{P}_1^L$ based on $\boldsymbol{t}_0$ and $\boldsymbol{r}_0$
  $s \leftarrow \text{size}(\boldsymbol{P}_1^J, \boldsymbol{M}_1)$
  $\boldsymbol{Q}_i^{\mathcal{L}} \leftarrow \frac{1}{s}\boldsymbol{P}_i^{\mathcal{L}} \forall i \in \{1,2\}$
  Initialize $\boldsymbol{s}$ and $\boldsymbol{u}$ from $\boldsymbol{T}_{\text{scaled}}^{\mathcal{L}\to\mathcal{J}}$
  repeat
    $\boldsymbol{Q}_i^{\mathcal{B}} \leftarrow \boldsymbol{T}_i^{\mathcal{J}\to\mathcal{B}} \circ \boldsymbol{Q}_i^{\mathcal{L}} \forall i \in \{1,2\}$
    $\boldsymbol{Q}_{\text{masked},i}^{\mathcal{B}} \leftarrow \text{removeMasked}(\boldsymbol{Q}_i^{\mathcal{B}}, \boldsymbol{M}_i) \forall i \in \{1,2\}$
    $\boldsymbol{D} \leftarrow \text{findMatches}(\boldsymbol{Q}_{\text{masked},1}^{\mathcal{B}}, \boldsymbol{Q}_{\text{filtered},2}^{\mathcal{B}})$
    $\boldsymbol{f} \leftarrow \text{filter}(\boldsymbol{D})$
    $\boldsymbol{D}_{\text{filtered}} \leftarrow \text{removeMasked}(\boldsymbol{D}, \boldsymbol{f})$
    $e \leftarrow$remaining error after solving for $\min_{s\in\boldsymbol{s},u\in\boldsymbol{u}}$
  until $|\Delta[e/\sum_{f\in\boldsymbol{f}} f]| \leq c$
  $\boldsymbol{T}^{\mathcal{L}\to\mathcal{J}} \leftarrow \boldsymbol{T}^{\mathcal{E}\to\mathcal{J}^{-1}}\text{sc}(\boldsymbol{T}_{\text{scaled}}^{\mathcal{L}\to\mathcal{J}}, s)$
  Compute $\boldsymbol{t}_{\text{final}}, \boldsymbol{r}_{\text{final}}$ from $\boldsymbol{T}^{\mathcal{L}\to\mathcal{J}}$
  return $(\boldsymbol{t}_{\text{final}}, \boldsymbol{r}_{\text{final}})^{\text{T}}$

directions. The results are saved to a mask $\boldsymbol{f}$ with

$$f_i = \max\left(\text{f}_1\left(a_i, b_i\right), \text{f}_2\left(a_i, b_i\right)\right). \tag{22}$$

We then define the cost for a match $\boldsymbol{d}$ by the point-to-plane distance as initially suggested by Chen and Medioni [11] and use the Levenberg-Marquard algorithm [20], [21] to find the optimal values for $\boldsymbol{s}$ and $\boldsymbol{u}$ to minimize the cost

$$e = \min_{s\in\boldsymbol{s},u\in\boldsymbol{u}} \sum_{\boldsymbol{d}\in\boldsymbol{D}} [(\boldsymbol{q}_b^{\mathcal{B}} - \boldsymbol{q}_a^{\mathcal{B}})\cdot \text{n}\left(\boldsymbol{q}_a^{\mathcal{B}}\right)]^2 | \boldsymbol{d} = \langle a, b\rangle. \tag{23}$$

We stop iterating once $|\Delta[e/\sum_{f\in\boldsymbol{f}} f]|$ reaches below a threshold c.

C. Retrieving the Extrinsic Parameters

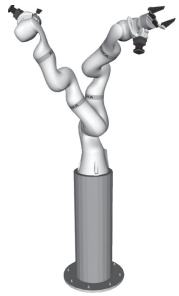
After the optimization loop we know $\boldsymbol{s}$ and $\boldsymbol{u}$ which parameterize $\boldsymbol{T}_{\text{scaled}}^{\mathcal{L}\to\mathcal{J}}$. However we are looking for $\boldsymbol{T}^{\mathcal{L}\to\mathcal{E}}$ which we can retrieve by

$$\boldsymbol{T}^{\mathcal{L}\to\mathcal{J}} = \boldsymbol{T}^{\mathcal{E}\to\mathcal{J}^{-1}}\text{sc}\left(\boldsymbol{T}_{\text{scaled}}^{\mathcal{L}\to\mathcal{J}}, s\right). \tag{24}$$

The final translation parameters $\boldsymbol{t}$ can now be read directly from the transformation matrix while the angles of $\boldsymbol{r}$ can be computed from its upper left rotational part.

IV. Evaluation

Our algorithm has been tested on three synthetic datasets and one real dataset recorded in our laboratory. The simulated environments used for the synthetic datasets contain cubic rooms with edge lengths of 5 m, 10 m and 20 m, similar to the ones used in [8]. The robot is mounted at a position of 25 % of the edge length in x and 33 % in y direction on a pillar 90 cm above the ground. The 3D scans have been recorded from the randomly chosen configurations shown in Fig. 5. We moved the robot joint with a speed of $0.1\,\text{rad}/\text{s}$ resulting in around 377,000 depth measurements during a 180° revolution.



Axis	1st Pose	2nd Pose
1	71°	−32°
2	6°	35°
3	−3°	117°
4	−46°	1°
5	10°	117°
6	26°	93°
7	−90° to 90°	−90° to 90°

Fig. 5. Robot scanning poses for synthetic datasets.

A. Convergence

We ran 100 tests on each dataset with a fixed number of iterations. For every run we started with a random error of up to 10 cm for each of x, y and z as well as up to 0.1 rad per Euler angle. As shown in figure 6 all runs converged, however the required number of iterations increase with the scale of the room. The shown translational offset is computed by the l^2 norm of the offset along x, y and z while the orientation error is the smallest possible angle between the combined rotation from α, β, γ and the ground truth. In larger room sizes we can also see quite common behavior for ICP implementations: A very short offset in any of the angular parameters can tremendously increase the cost for far distant points, resulting in excessive influence of those parameters during the optimization. On the 20 m cube dataset we can see how the angular offset is continuously decreasing while the translational offset is even increasing for some of the first iterations. However once the angular error is under control the linear parameters converge as well.

We ran a similar test with two real 3D scans of our laboratory. To decrease the influence of distant points we cropped our point clouds by masking all points with $\|\boldsymbol{p}^{\mathcal{B}}\|$ greater than 10 m. The results in figure 7 show that the algorithm also converges on complex, real-data but requires more iterations to do so. We compared our results to transformation calculated from the CAD data of our the 3d-printed sensor mount. However we need to stress out that these values might also be incorrect due to tolerances in production and assembly.

Our calibration converges to a solution with average offsets of 21.1 mm in the estimated position and 0.012 rad (0.66°) on the orientation parameters. A direct comparison the 3D data generated based on our results and parameters taken from CAD can be found in figure 1.

B. Precision

To evaluate the precision of our method we re-ran it on our synthetic datasets with different ground truth values for $\boldsymbol{T}^{\mathcal{L}\to\mathcal{E}}$ as shown in figure 9. Each combination has been tested with added Gaussian noise with a standard deviation of $\sigma = 0.018$ m to our simulated depth measurements according to [22] as well as without sensor noise. The initial guesses have been initialized

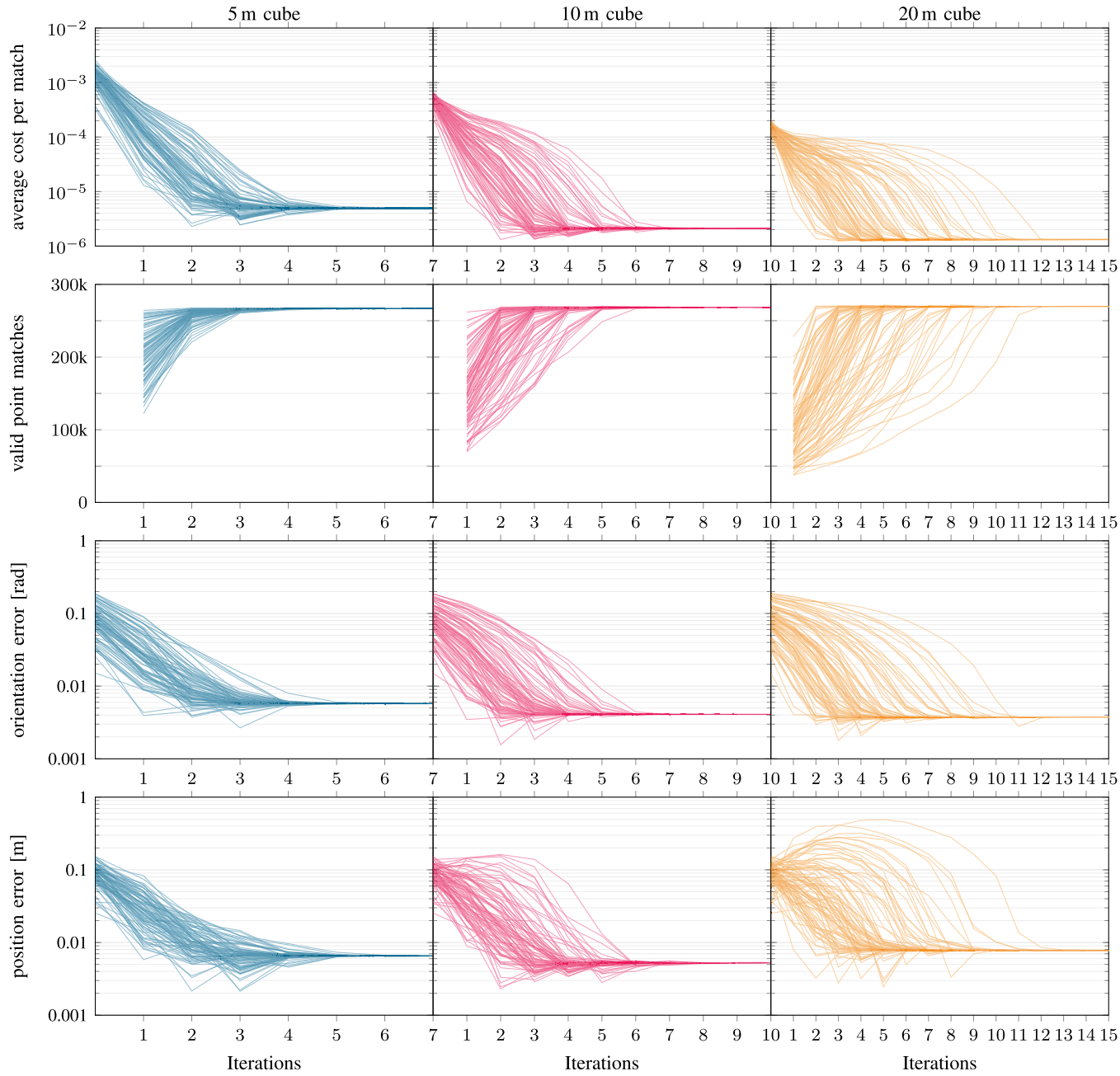


Fig. 6. Development of the number of point matches and their average cost as well as orientation and position error over multiple iterations. For each dataset 100 test runs with randomized offsets on the initial guess have been performed. The values are taken after every iteration. The average cost for iteration 0 is based on the matches of the first iteration before any optimization has been performed. The errors for iteration 0 show the initial guess.

by adding randomized offsets to the ground truth values as in the previous experiments. We performed twenty test runs for every scenario.

Overall we achieved an average position error of 10.6 mm and 0.006 rad (0.34°) of rotational offset with added noise and 7.3, mm and 0.005 rad (0.29°) without. The worst case results have been on the parameters set $\boldsymbol{c}_1$ in the 5 m cube dataset with noise on a position error of 25.7 mm and an orientation error of 0.011 rad (0.64°). The precision of our results is comparable to the ones shown in [8] (worst case 23.7 mm position and 0.55° rotation error; no averages given) even though they calibrated only four degrees of freedom and used a noticeably lower noise ration of $\sigma = 0.01$ m.

Moreover the tests show that the effect of sensor noise on the calibration result decreases with the size of the scanned scene. Without noise there is almost no variance in the results causing the boxes plotted in figure 8 to appear like bars.

C. Runtime

As shown infigure 10 the calibration needed on average 6.2 iterations and 200.9 s to find a solution on the synthetic

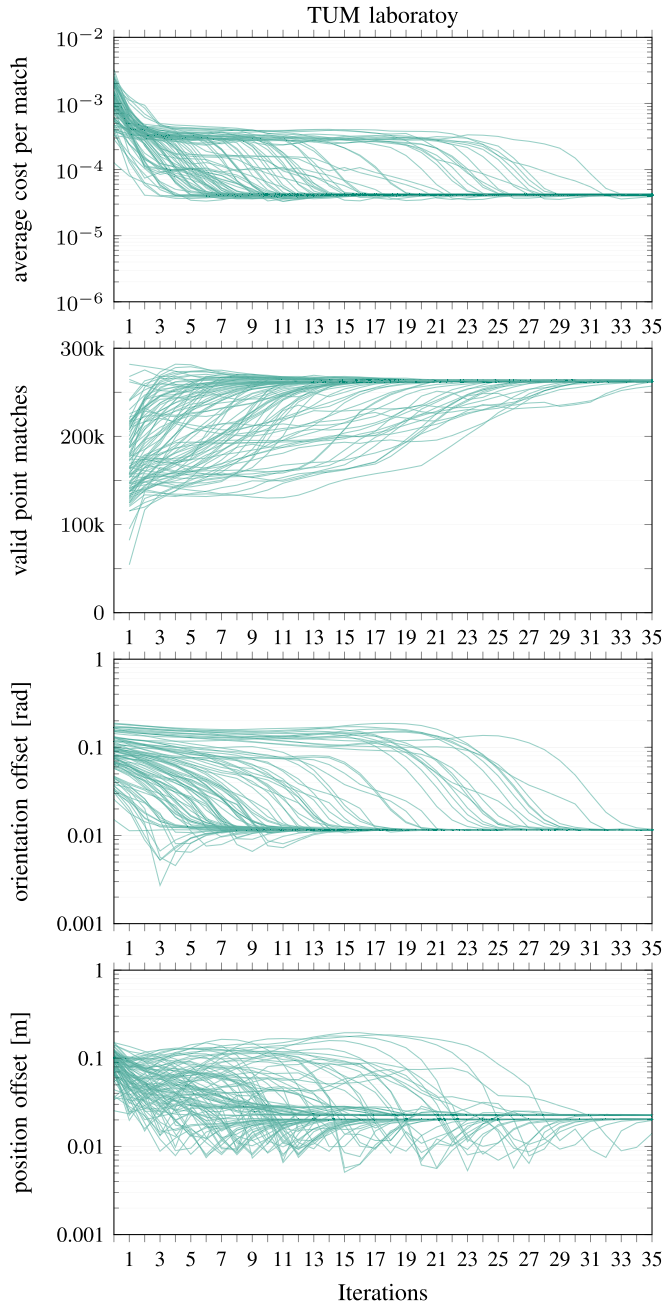


Fig. 7. Convergence of calibration tested with two recordings of our laboratory. The orientation and position errors have been calculated by comparing our pose estimation to the CAD data of the sensor mount. Iteration 0 shows the initial guess. The average cost for iteration 0 is based on the matches of the first iteration before performing optimization.

datasets without noise and 14.2 iterations in 408.2 s with noise. It is also visible that the runtime of the calibration on noisy data decreases the larger the room gets: It went down from 17.6 iterations (485.2 s) in the 5 m room over 12.8 (399.0 s) in the 10 m environment to 12.2 iterations (340.3 s) on the 20 m room datasets. One can also see that the number of point matches has a measurable influence on the overall runtime: The test runs with the mounting configuration $\boldsymbol{c}_3$ in which the robot arm

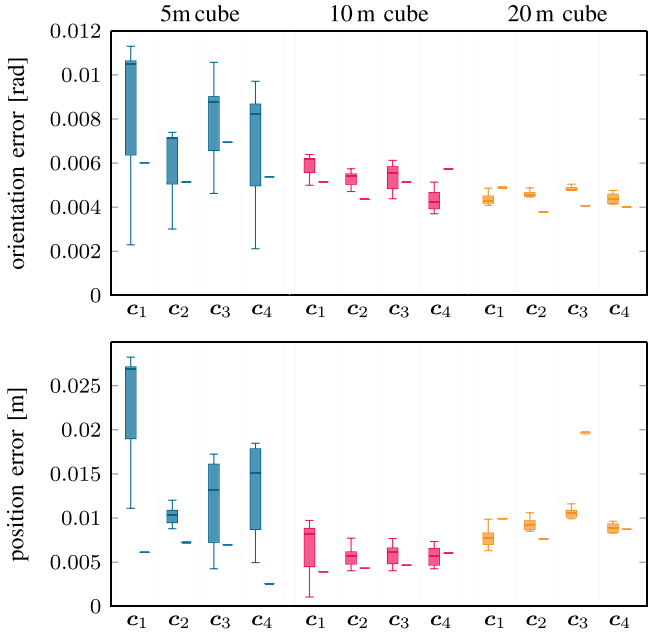


Fig. 8. Distribution of errors for multiple datasets. Every distribution shown is based on 20 calibration runs with randomized initial guesses. We tested three simulation environments (colors) with four different ground truth parameters (pairs), each with and without noise (left entries per column are with noise, right ones without).

Name	x	y	z	α	β	γ
$\boldsymbol{c}_1$	6 mm	0 mm	-139 mm	1.571 rad	0.000 rad	1.571 rad
$\boldsymbol{c}_2$	-75 mm	-56 mm	-175 mm	1.536,rad	-0.054 rad	1.471 rad
$\boldsymbol{c}_3$	101 mm	29 mm	-144 mm	1.531 rad	-0.021 rad	1.541 rad
$\boldsymbol{c}_4$	-79 mm	68 mm	-237 mm	1.591 rad	-0.001 rad	1.601 rad

Fig. 9. Parameter vectors $\boldsymbol{c}_i$ used as ground truth for $\boldsymbol{T}^{\mathcal{J}\to\mathcal{E}}$. $\boldsymbol{c}_1$ is based on CAD model of our robot, $\boldsymbol{c}_2$ to $\boldsymbol{c}_4$ are based on $\boldsymbol{c}_1$ with random offsets.

Scene	Conf.	Noise	Av. Iterations	Av. Matches / Iteration	Av. Runtime
5 m cube	$\boldsymbol{c}_1$	no	5.6	254,306.2	214.0 s
		yes	17.2	178,209.1	471.3 s
	$\boldsymbol{c}_2$	no	6.0	257,346.7	212.2 s
		yes	17.8	200,140.1	542.6 s
	$\boldsymbol{c}_3$	no	5.5	190,081.9	141.1 s
		yes	17.3	150,933.7	393.1 s
	$\boldsymbol{c}_4$	no	5.5	253,783.4	194.9 s
		yes	18.0	201,220.0	533.6 s
10 m cube	$\boldsymbol{c}_1$	no	6.2	233,495.0	207.4 s
		yes	14.2	207,814.1	469.2 s
	$\boldsymbol{c}_2$	no	5.9	236,535.8	201.9 s
		yes	12.2	207,045.1	392.4 s
	$\boldsymbol{c}_3$	no	6.0	185,081.9	169.0 s
		yes	11.7	162,567.0	294.0 s
	$\boldsymbol{c}_4$	no	6.2	238,589.8	231.9 s
		yes	13.1	215,346.0	440.6 s
20 m cube	$\boldsymbol{c}_1$	no	6.9	187,562.3	218.5 s
		yes	13.4	171,004.5	386.8 s
	$\boldsymbol{c}_2$	no	6.9	202,143.8	227.9 s
		yes	12.0	174,548.2	338.4 s
	$\boldsymbol{c}_3$	no	6.7	157,300.5	175.6 s
		yes	10.3	137,774.5	234.8 s
	$\boldsymbol{c}_4$	no	6.9	198,710.9	216.4 s
		yes	13.1	181,965.9	401.4 s
Laboratory	$\boldsymbol{c}_1$	yes	19.6	243,519.2	685.0 s

Fig. 10. Average number of iterations, average number of valid points matches per iteration and average runtime per calibration, each based on 20 runs per row.

occludes large parts of the room were the fastest ones in every environment.

The calibration on real data took on average 19.6 iterations with a runtime of 685.0 s. All runtime benchmarks have been performed by running a single core implementation on a Dell XPS 15 9550 notebook (3rd generation from 2015) featuring an Intel Core i7 6700HQ CPU and 16 GB of RAM.

V. Conclusion and Future Work

We presented an easy to use method for calibrating the extrinsic pose of a 2D rangefinder mounted to a robot arm. The approach works in unstructured environments without requiring any external calibration targets, though complicated environments may increase the computing time as well as contain local minima. It converges reliably—even with crude initial guesses—and gives precise results for all six degrees of freedom, comparable to other state-of-the art approaches that only work on a reduced 4 DOF complexity. As such it allows an on-site self-calibration of eye-in-hand line-based depth sensors to robotic actuators.

In our future work we plan to further investigate the time-wise synchronization of sensor and actor. We are also planning on testing our calibration method for other scanning patterns and sensor types such as depth cameras and/or triangulation based laser scanners. Even though filtering of invalid point matches already allows to compensate for minor changes in the scene the next challenge would be to enable calibration in dynamic environments. Last but not least we are preparing a benchmark to retrieve comparable results with some of the calibration procedures discussed in section II. Even though we are still working on the benchmark our first results indicate a correlation between the number of measurements taken into account by a calibration method and the precision of the results: Traditional methods relying on only a few measurement points on a specific calibration target are very prone to sensor noise and get clearly outperformed by bundle adjustment approaches like [5], [8] and ours.

Acknowledgment and Implementation Details

Great thanks to Savatore Virga and Marco Esposito for their work on the iiwa_stack project [23] which has been used for our test on real hardware as well as in simulation. On top we improved the rosjava support for ROS actions as well as the support for applying velocity constraints for the KUKA LBR iiwa and ported TF to java. All of those developments have been integrated to the according projects; our implementation of rosjava_tf can be found as open source at [24]. Our synthetic datasets have been generated with Gazebo while our optimization is based on Ceres [25]. Also many thanks to Dinesh Paudel who was a great help in the construction process of our robot workcell.

References

[1] M. Wagner, P. He, S. Reitelshfer, and J. Franke, " Self-calibration method for a robotic based 3D scanning system," in *Proc. IEEE Int. Conf. Emerg. Technologies Factory Autom.*, 2015, pp. 1–6.
[2] S. Chen *et al.*, "Extrinsic calibration of 2D laser rangefinders based on a mobile sphere," *Remote Sens.*, vol. 10, no. 8, 2018, Art. no. 1176.
[3] M. E. Antone and Y. Friedman, " Fully automated laser range calibration," in *Proc. British Mach. Vision Conf.*, 2007, pp. 66-1–66-10.
[4] T. T. Andersen, N. A. Andersen, and O. Ravn, "Calibration between a laser range scanner and an industrial robot manipulator," in *Proc., IEEE Symp. Comput. Intell. Control Autom.*, 2014, pp. 1–8.
[5] V. Pradeep, K. Konolige, and E. Berger, "Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach," in *Experimental Robotics*. Berlin, Germany: Springer, 2014, pp. 211–225.
[6] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2007, pp. 4164–4169.
[7] M. Sheehan, A. Harrison, and P. Newman, "Automatic self-calibration of a full field-of-view 3D n-laser scanner," in *Experimental Robotics*. Berlin, Germany: Springer, 2014, pp. 165–178.
[8] J. Oberländer, L. Pfotzer, A. Roennau, and R. Dillmann, "Fast calibration of rotating and swivelling 3-D laser scanners exploiting measurement redundancies," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 3038–3044.
[9] T. Yoshida, K. Irie, E. Koyanagi, and M. Tomono, "3D laser scanner with gazing ability," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3098–3103.
[10] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *Proc. SPIE*, vol. 1611, pp. 586–606, 1992.
[11] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vision Comput.*, vol. 10, no. 3, pp. 145–155, 1992.
[12] J. Kelly and G. S. Sukhatme, "A general framework for temporal calibration of multiple proprioceptive and exteroceptive sensors," in *Experimental Robotics*. Berlin, Germany: Springer, 2014, pp. 195–209.
[13] Y. Huang, X. Qian, and S. Chen, "Multi-sensor calibration through iterative registration and fusion," *Comput.-Aided Design*, vol. 41, no. 4, pp. 240–255, 2009.
[14] R. S. Yang *et al.*, "Multi-kinect scene reconstruction: Calibration and depth inconsistencies," in *Proc. IEEE 28th Int. Conf. Image Vision Comput. New Zealand (IVCNZ)*, 2013, pp. 47–52.
[15] N. Heide, T. Emter, and J. Petereit, "Calibration of multiple 3D LiDAR sensors to a common vehicle frame," in *Proc. 50th Int. Symp. Robot.*, 2018, pp. 1–8.
[16] H. Alismail, L. D. Baker, and B. Browning, "Automatic calibration of a range sensor and camera system," in *Proc. 2nd IEEE Int. Conf. 3D Imag., Model., Process., Visualization Transmiss.*, 2012, pp. 286–292.
[17] H. Alismail and B. Browning, "Automatic calibration of spinning actuated LiDAR internal parameters," *J. Field Robot.*, vol. 32, no. 5, pp. 723–747, 2015.
[18] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[19] R. A. Newcombe *et al.*, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, 2011, vol. 11, no. 2011, pp. 127–136.
[20] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math.*, vol. 2, no. 2, pp. 164–168, 1944.
[21] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, 1963.
[22] F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, and R. Siegwart, "Noise characterization of depth sensors for surface inspections," in *Proc. IEEE 2nd Int. Conf. Appl. Robot. Power Industry*, 2012, pp. 16–21.
[23] C. Hennersperger *et al.*, "Towards MRI-based autonomous robotic us acquisitions: A first feasibility study," *IEEE Trans. Med. Imag.*, vol. 36, no. 2, pp. 538–548, Feb. 2017.
[24] A. Peters and N. A. Crews, "rosjava_tf," 2019. [Online]. Available: https://github.com/exo-core/rosjava_tf
[25] S. Agarwal, K. Mierle, and Others, "Ceres solver," 2018. [Online]. Available: http://ceres-solver.org.