Received: 4 October 2022 | Revised: 5 September 2023 | Accepted: 2 October 2023

DOI: 10.1002/rob.22259

RESEARCH ARTICLE

WILEY

Robot self-calibration using actuated 3D sensors

Arne Peters[1,2] | Alois C. Knoll[1]

[1]Chair of Robotics, Artificial Intelligence and Real-time Systems, TUM School of Computation, Information and Technology, Technical University of Munich (TUM), Garching bei München, Germany

[2]Infinisense Technologies GmbH, Munich, Germany

Correspondence

Arne Peters, Infinisense Technologies GmbH, Einsteinstr. 174, Munich 81677, Germany.
Email: arne.peters@tum.de and peters@infinisense.de

Funding information

European Union's Horizon 2020 Research and Innovation Programme,
Grant/Award Number: 870133

Abstract

Both robot and hand-eye calibration have been object of research for decades. While current approaches manage to precisely and robustly identify the parameters of a robot's kinematic model, they still rely on external devices such as calibration objects, markers and/or external sensors. Instead of trying to fit recorded measurements to a model of a known object, this paper treats robot calibration as an offline SLAM problem, where scanning poses are linked to a fixed point in space via a moving kinematic chain. As such, we enable robot calibration by using nothing but an arbitrary eye-in-hand depth sensor. To the authors' best knowledge the presented framework is the first solution to three-dimensional (3D) sensor-based robot calibration that does not require external sensors nor reference objects. Our novel approach utilizes a modified version of the Iterative Corresponding Point algorithm to run bundle adjustment on multiple 3D recordings estimating the optimal parameters of the kinematic model. A detailed evaluation of the system is shown on a real robot with various attached 3D sensors. The presented results show that the system reaches precision comparable to a dedicated external tracking system at a fraction of its cost.

KEYWORDS

3D reconstruction, calibration and identification, range sensing, SLAM

1 | INTRODUCTION

In 2018 the American Automobile Association published a report about the repair cost of modern cars with *Advanced Driver Assistance Systems* (ADAS), indicating that repairs are two to three times more expensive than for traditional cars (American Automobile Association, Inc, 2018). These additional costs are the result of both having to replace modern integrated sensors, as well as the need for their calibration, requiring not only dedicated equipment but also specially trained personnel. In a similar manner Richardson et al. (2013) performed a survey on camera calibration, comparing the achieved precision reached by laymen and experts. Their findings confirm the necessity for qualified personnel as the calibration quality heavily depends on capturing sufficient and evenly distributed footage of the used calibration object over the entire image area. While there are no similar studies available for robot calibration, it is reasonable to assume that the effects with respect to cost and required know-how are similar.

The biggest drawback of current calibration techniques is, however, that they rely on specialized equipment. While many

Abbreviations: 3D, three-dimensional; ADAS, advanced driver assistance systems; CAD, computer-aided design; CMM, contact measuring machine; CPC model, complete and parametrically continuous model; DH, Denavit-Hartenberg; EE, end-effector; ICP algorithm, Iterative Corresponding Point algorithm; LiDAR, light detection and ranging; MCPC model, modified complete and parametrically continuous model; NFOV, narrow field of view; PSO-GP, particle swarm optimization-Gaussian process; SL, structured light; SLAM, simultaneous localization and mapping; ToF, time-of-flight; TUM, Technical University of Munich; URDF, unified robot description format; WFOV, wide field of view.

Note that affiliation 2 has no involvement with this publication at all. The entire paper was written and submitted while author was still employed at TUM.

approaches from literature are presented as "autonomous" or "automated," they can only be used in conjunction with additional, manually placed calibration objects, markers, and/or sensors, that is, dot (Zhang et al., 2017) or checkerboard patterns (Pradeep et al., 2014; Strobl & Hirzinger, 2006; Tsai & Lenz, 1989b), spheres (Bi et al., 2017; Chen et al., 2018; Knoll, n.d; Li et al., 2008; Wang et al., 2020; Yin et al., 2014; Yu & Xi, 2018) or other specifically designed calibration targets (Andersen et al., 2014; Antone & Friedman, 2007). Thus, already deployed systems in unpredictable environments (e.g., city traffic, households, emergency scenarios, etc.) are rendered impossible to recalibrate on-site. Though calibration often wrongly treated as a once-in-a-lifetime action, system parameters are changing over a system's usage period, due to inevitable environmental factors such as wear-and-tear, maintenance and repairs, changes in temperature or mechanical stress, that is, caused by shipping or collisions. The consequences of wrong parameters may range from small imprecisions, over task failure, up to a potential loss of an entire robotic system, when deployed to a hazardous environment of which it cannot escape by itself anymore.

To overcome the aforementioned issues, this paper presents a framework allowing true on-site self-calibration of a robot system equipped with an arbitrary eye-in-hand 3D sensor. Our method enables a robot to recalibrate itself in any unknown, static environment. For example, an end-user could leave his bent service alone in the bedroom for an hour after a minor incident and it becomes fully operational again, or an exploration robot may use a few scans of a random rock to restore its full functionality, after taking a hit or landing a foreign body.

Instead of using external utilities it is based on point cloud registration techniques to fuse multiple scans of a given scene as shown in Figure 1. We extend the *Iterative Corresponding Point* (ICP) algorithm-based formulations for extrinsic senor calibration (six parameters) presented in Peters et al. (2020) to find the optimal parameters of an entire kinematic chain. This includes all calibratable parameters of a robot manipulator as well as the hand-to-eye transformation—a total of $4r + 2p$ parameters, where r is the number of revolute and p the number of prismatic joints ($4 \times 7 = 28$ for the shown KUKA LBR iiwa robot arm). In summary, the key-contributions of this article are:

1. To the best of the authors' knowledge, this is the first approach solving the calibration of an entire robot system by relying only on depth data instead of external tools and objects.
2. By introducing bundle-adjustment on more than just one pair of scans, it becomes possible to calibrate more than just six parameters while reducing the risk of local minima and improving the overall calibration precision.
3. The presented framework allows calibration of any kinematic chain and depth sensor combination, for example, single beam LiDARs, line scanners, and depth cameras.
4. A detailed evaluation is presented, comparing multiple real-world hardware configurations to a calibration performed using traditional methods with a dedicated three-dimensional (3D) tracking system.

2 | STRUCTURE

Our article is structured as follows: Section 3 starts by giving an overview on the mathematical modeling of kinematic chains and introduces the *modified complete and parametrically continuous* (M-CPC) model (Zhuang et al., 1992). From there we turn toward more recent, related work, covering the states-of-art in 3D scan registration as well as robot calibration. In Section 5, we then introduce our own framework for target-less calibration. The section basically follows our processing pipeline, starting from the data pre-processing and finishing with the final error metric to be optimized. A detailed evaluation of our approach is given in Section 6. We show a convergence analysis of our algorithm on synthetic data and measure its precision by comparing it to a reference calibration obtained by a professional 3D tracking system. Finally, a summary is given in Section 7.

3 | FUNDAMENTALS

Today, various technologies for measuring 3D information can be found in the market. The most common ways to contactless estimate the distance between a sensor and a surface, are stereo vision (Cyganek & Siebert, 2011; Ma et al., 2012; Sturm & Ramalingam, 2011),

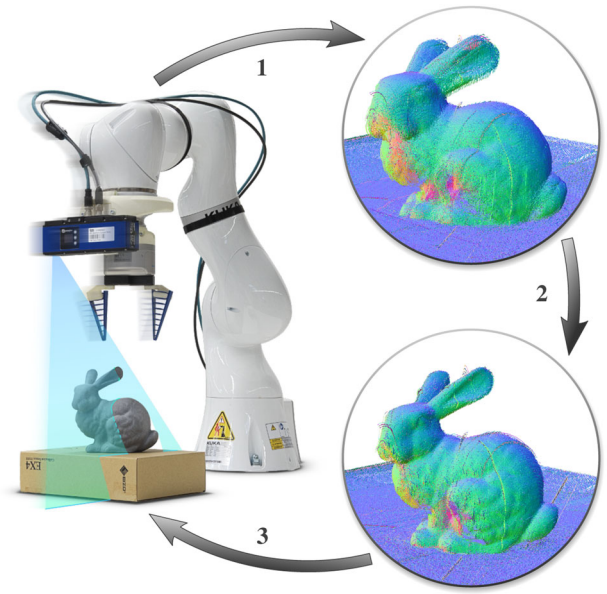

FIGURE 1 Visualization of the proposed calibration pipeline: (1) A robot with an attached 3D sensor captures multiple recordings of an arbitrary scene by moving the eye-in-hand sensor. (2) The 3D scans transformed to the coordinate frame of the robot's base by using its kinematic model. We apply Iterative Corresponding Point-based bundle adjustment to align and undistort the point clouds through optimization the model's parameters. Note how the crispness improves on the bunny's ear and crib. (3) The obtained calibration parameters are applied to the robot manipulator.

structured light (SL) (Zhang, 2018) and the *time-of-flight* (ToF) principle (Hansard et al., 2012), with some technologies allowing for multiple simultaneous measurements. According to the used sensor model, the range or depth measurements captured at moment t can be projected to a set of points

$$P_t = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\} \tag{1}$$

with

$$\boldsymbol{p}_i = (x_i, y_i, z_i)^T. \tag{2}$$

Each $\boldsymbol{p}_i$ is a 3D point relative to the sensor's origin. As the sensor is assumed to be mounted in an eye-in-hand configuration, its origin also forms the *end effector* (EE) frame $\mathcal{E}$ of the robot's kinematic chain. In the following work, coordinate frames of points will be denoted in superscript, calligraphic letters, such as in $\boldsymbol{p}_i^{\mathcal{E}}$.

When actuating the robot, the relative pose of $\mathcal{E}$ to the robot's base $\mathcal{B}$ changes. Thus, point clouds taken from different robot configurations do not align anymore. One possible strategy for fusing multiple point clouds $P_0^{\mathcal{E}}, \ldots, P_n^{\mathcal{E}}$ is to transform them to the static coordinate frame $\mathcal{B}$. The required transformation

$$\boldsymbol{T}_t^{\mathcal{E}\to\mathcal{B}} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \tag{3}$$

can be computed from the kinematic parameters $\boldsymbol{k}$ of the robot and its joint states $\boldsymbol{j}_t$ at the scan recording time t:

$$\boldsymbol{T}_t^{\mathcal{E}\to\mathcal{B}} = \mathrm{tr}(\boldsymbol{k}, \boldsymbol{j}_t), \tag{4}$$

so that

$$\begin{pmatrix} \boldsymbol{p}_i^{\mathcal{B}} \\ 1 \end{pmatrix} = \mathrm{tr}(\boldsymbol{k}, \boldsymbol{j}_t) \begin{pmatrix} \boldsymbol{p}_i^{\mathcal{E}} \\ 1 \end{pmatrix} \quad \forall\, \boldsymbol{p}_i^{\mathcal{E}} \in P_t^{\mathcal{E}}. \tag{5}$$

While $\boldsymbol{j}_t$ is simply the sequence of joint positions along the robot from base to EE as

$$\boldsymbol{j}_t = (j_{1,t}, j_{2,t}, \ldots, j_{n,t})^T, \tag{6}$$

the definition of $\boldsymbol{k}$ is not as straightforward: A kinematic model suitable for calibration should be continuous and represent a complete set of *degrees of freedom* (DoF) while staying nonredundant. Despite its age, the probably most common manner of representing robot manipulator kinematics is still the *Denavit–Hartenberg* (DH) convention (Denavit & Hartenberg, 1955). As modeling a kinematic chain with six-DoF per transformation includes several redundancies (e.g., one could place a revolute joint anywhere along its rotation axis and still obtain the same EE pose) the DH convention defines joints to move along or around their local z-axis, while using only four parameters ϕ_n, d_n, a_n, and α_n per segment n, where ϕ_n is the rotation around z_{n-1}, d_n the translation along z_{n-1}, a_n the translation along x_n and α_n the rotation around x_n. z_{n-1} is the z-axis of the prior joint in the kinematics chain.

Unfortunately, the DH model suffers from multiple drawbacks: It is neither complete nor parametrically continuous. To overcome these issues Stone (1987) suggested to use two additional parameters b_n and γ_n per segment in his *S-model*, making the model complete, but not parametrically continuous at the cost of introducing redundancies.

Zhuang et al. presented an alternative approach, the *Complete and Parametrically Continuous* (CPC) model (Zhuang et al., 1992), later refined to the more intuitive *Modified-CPC* (MCPC) model (Zhuang et al., 1993). Similar to the DH convention the MCPC model expects all joints move along or around their local z-axes. It is constructed for each segment of the robot's kinematic chain, connecting two joints $\mathcal{J}_i$ and $\mathcal{J}_{i+1}$, by rotating the frame $\mathcal{J}_i$ around its x and y axes to align its xy-plane with the one of $\mathcal{J}_{i+1}$ and then shift it along the new x and y axes to position the new origin on the z-axis of $\mathcal{J}_{i+1}$.

The MCPC model defines $\mathrm{tr}(\boldsymbol{k}, \boldsymbol{j}_t)$ as a product of transformations, alternating between static segment and variable joint transforms:

$$\mathrm{tr}(\boldsymbol{k}, \boldsymbol{j}_t) = \mathrm{st}\left(\boldsymbol{s}_0^T\right) \cdot j_t(j_1) \cdot \mathrm{st}\left(\boldsymbol{s}_1^T\right) \cdot \cdots \cdot j_t(j_n) \cdot \mathrm{st}\left(\boldsymbol{s}_n^T\right) \tag{7}$$

with

$$\boldsymbol{k} = \left(\boldsymbol{s}_0^T, \boldsymbol{s}_1^T, \boldsymbol{s}_2^T, \ldots, \boldsymbol{s}_n^T\right)^T, \tag{8}$$

where $\boldsymbol{s}_i$ are the static parameters of the ith joint. The MCPC model uses a total of four parameters per revolute joint, two per prismatic joint and six-DoF for the transformation between the last joint and the robot's EE. For revolute joints, each segment is defined by $\boldsymbol{s} = (\alpha, \beta, x, y)^T$ so that:

$$\mathrm{st}(\boldsymbol{s}^T) = \mathrm{rot}(\boldsymbol{u}_x, \alpha) \cdot \mathrm{rot}(\boldsymbol{u}_y, \beta) \cdot \mathrm{trans}(x, y, 0), \tag{9}$$

where $\mathrm{rot}(\boldsymbol{a}, \alpha)$ is a rotation of α around axis $\boldsymbol{a}$ and $\mathrm{trans}(x, y, z)$ a translation along $(x, y, z)^T$. $\boldsymbol{u}_x$ denotes the unit vector of a coordinate frames local x-axis; $\boldsymbol{u}_y$ and $\boldsymbol{u}_z$ for the y and z axes accordingly. For prismatic joints the parameters x_i and y_i are treated as zero.

One special case is the transformation between the last joint and the EE, which includes two additional degrees of freedom γ and z, so that

$$\mathrm{st}\left(\boldsymbol{s}_n^T\right) = \mathrm{rot}(\boldsymbol{u}_x, \alpha_n) \cdot \mathrm{rot}(\boldsymbol{u}_y, \beta_n) \cdot \mathrm{rot}(\boldsymbol{u}_z, \gamma) \cdot \mathrm{trans}(x_n, y_n, z). \tag{10}$$

4 | RELATED WORK

The presented approach combines techniques from two different fields of research: (1) Point cloud registration, which is commonly used in computer vision, for example, for 3D reconstruction and/or *simultaneous localization and mapping* (SLAM) as well as (2) solving the hand-eye and/or robot calibration problems, where especially the second one is common the field of control engineering. As such the state of the art in both fields will be presented separately.

4.1 | Point cloud registration

Regardless of its 30th anniversary, the widest-known approach for aligning two point clouds is still the ICP algorithm, developed independently by Besl and McKay (1992) as well as Chen and Medioni (1992) in the same year. It aims to find the rigid transformation required to align one point cloud ("data") with a second one ("model"), by iteratively searching pairs of closest points between both clouds and optimizing the initially guessed transformation by minimizing the distance of all pairs. While the general idea of both ICP versions is the same, Besl and McKay used the squared Cartesian distance of matching points as an error measure, while Chen and Medioni optimized the point-to-plane distance, which was later shown to reach a faster convergence (Rusinkiewicz & Levoy, 2001).

Over the last decades numerous variants of the ICP algorithm have been developed, employing various strategies for point matching, introducing an additional validation step for point matches, and/or varying metrics as well as optimization techniques. Detailed overviews of shape matching are given in the survey papers (Diez et al., 2015; Pomerleau et al., 2015). An older survey of Rusinkiewicz and Levoy (2001) even performed a benchmark of different ICP variants. He also suggests that *ICP* might be a better fit for the ICP acronym, since many other matching criteria (e.g., features or backprojection) other than pure geometrical distance have been shown.

Two interesting and more recent extensions of the ICP algorithm come from Segal et al. (2009), which formulated a plane-to-plane distance function, as well as from Rusinkiewicz (2019) introducing a symmetric objective cost metric. Other recent works try to extend the ICP algorithm to support non-rigid shape matching (Amberg et al., 2007; Brown & Rusinkiewicz, 2007; Cheng et al., 2017) or get rid of the required initial guess (Attia & Slama, 2017; Cop et al., 2018). Furthermore, researchers have applied deep learning to one or multiple steps of the algorithm (Wang & Solomon, 2019a), as well as to solve the problem of point cloud registration solely by machine learning (Wang & Solomon, 2019b; Yu et al., 2021).

4.2 | Calibration

Calibrating a robot system with a hand and an eye can be broken down to three separate problems: (1) Intrinsic calibration of the optical sensor, (2) finding the transformation between the actuator and the eye, and (3) calibration of the actuator itself. This partitioning was coined by Tsai and Lenz, who presented a series of papers solving each step separately (Tsai & Lenz, 1989a).

4.2.1 | Sensor calibration

The issue of sensor calibration is naturally depending on the sensor and commonly treated as a standalone problem. Even works combining approaches for solving entire system calibration at once usually treat sensor calibration as an independent step in the overall calibration pipeline (such as Birbach et al., 2015; Fuchs & Hirzinger, 2008; Miseikis et al., 2016).

Naturally, a suitable sensor calibration must be selected for the specific device used. Camera calibration in particular has become a standalone research field. An early approach to classic camera calibration was the *Eight-point* algorithm (Longuet-Higgins, 1981), which only became stable after introducing an additional normalization procedure (Hartley, 1997). Another widely used calibration approach is the method of Zhang (2000). Later works introduced more extensive sensor models, that is, by including additional parameters for modeling radial distortion (Kannala & Brandt, 2006; Wang et al., 2008).

Also, in the context of 3D perception more complex sensor models are required. For stereo vision systems, the camera parameters of both sensors as well as the calibration between them, needs to be known, while in SL and ToF systems one must consider the projector instead of a second camera. In Fuchs and Hirzinger (2008), the latency of a ToF sensor's projector is calibrated and Yamazoe et al. (2012) demonstrate that the projector of a Kinect v1 sensor can be modeled and calibrated in a similar fashion as a camera. Finally, Muhammad and Lacroix (2010) and Atanacio-Jiménez et al. (2011) investigate the calibration of rotation multibeam LiDAR sensors.

4.2.2 | Eye-to-hand calibration

In contrast to sensor calibration, the problems of robot calibration and eye-to-hand calibration are strongly coupled. While there are many works assuming an already calibrated actuator, consequently focusing only on finding the transformation between sensor and robot (e.g., Andersen et al., 2014, 1999; Antone & Friedman, 2007; Bi et al., 2017; Chen et al., 2018; Heller et al., 2015; Li et al., 2008; Tsai & Lenz, 1989b; Wagner et al., 2015; Yin et al., 2014; Zhang et al., 2017), calibration approaches for an entire robot often include the eye-to-hand transformation as just another robot segment. All aforementioned calibration methods for eye-to-hand calibration further rely on dedicated calibration objects, such as dot (Zhang et al., 2017) or checkerboard patterns (Tsai & Lenz, 1989b), spheres (Bi et al., 2017; Chen et al., 2018; Li et al., 2008; Yin et al., 2014), a pin (Wagner et al., 2015), as well as specifically deigned calibration targets such as a pyramid (Antone & Friedman, 2007) or a board with a cut-out triangle (Andersen et al., 2014).

More recent works tried to remove the requirement for special targets: Carlson et al. (2015) calibrated the eye-to-hand transformation by measuring generic planes, while Xu et al. (2022) used straight edges of random objects. Heide et al. (2018) estimated the pose of external LiDAR scanners by detecting a CAD-based 3D model of an excavators arm in the recorded point clouds. Sheehan et al. (2014) managed to intrinsically self-calibrate a multi-LiDAR scanning device without prior knowledge of the environment or other requirements

toward it. They defined a *crispness* error metric based on squared Renyi entropy, optimizing multiple overlying scans for crisp edges. In Alismail et al. (2012) and Alismail and Browning (2015), Alismail et al. solved the intrinsic calibration (four-DoF) of a self-built 3D LiDAR made from a rotation 2D scanner, by applying ICP on data from the first and second half of a single rotation. We recently showed that extrinsic calibration of an eye-in-hand LiDAR is possible via fusing two 3D scans taken at random manipulator configurations by rotating the wrist joint (Peters et al., 2020), which finally led to the question of whether ICP would also be suitable for calibration of an entire robot. Another idea for target-less eye-to-hand calibration was later presented by Li et al., where *the particle swarm optimization-Gaussian process* (PSO-GP) was used to fuse data of a close-range eye-in-hand line scanner, as ICP "is difficult to directly apply to the calibration of line laser sensors because the line laser sensors do not have enough scanning range" (Li, Du, et al., 2021, p. 2)—Quite in contrary to the findings of this paper: The experimental observations presented in Section 6 actually indicate, that the calibration precision is even higher when using ICP on smaller and less complex scenes.

4.2.3 | Robot calibration

While target-less eye-to-hand calibration of depth sensors has been demonstrated, there are no standalone solutions for entire robot calibration yet. General overviews to the problem of robot calibration are given in Abderrahim et al. (2004) and Chen-Gang et al. (2014). Examples for pure robot calibration are rather rare: Bennett and Hollerbach (1991) create a loop closure in a kinematic chain, by connecting two robots of the same type at their EEs, while Lightcap et al. (2008) and Mustafa et al. (2008) use a *Contact Measuring Machine* (CMM) and a precisely manufactured reference fixture to measure the position of the EE.

Another series of approaches is using external, optical measuring systems. These works usually treat the transformation between the EE and the sensor and/or markers as yet another segment of the unknown kinematic chain and include it in the calibration problem, for example, by using a theodolite and a reflector mounted to the robot's EE (Judd & Knasinski, 1990) or camera/marker based 3D tracking systems (Jang et al., 2001; Özgüner et al., 2020). Birbach et al. (2015) calibrate the manipulator of a humanoid-like robot by watching a marker on its wrist. Maye et al. (2016) demonstrated a first makerless solution by using deep learning to guess both a manipulator's kinematic model and configuration from watching it with an external camera.

Finally, a number of works solve robot calibration by using eye-in-hand devices. These methods tend to follow a similar approach as for eye-to-hand calibration. A calibration target is recorded from a number of manipulator poses. Using the constraints that neither the target, nor the robot's base have moved it is possible to to calculate the optimal parameters for the kinematic model. Even the used targets are often the same as those used for eye-in-hand calibration: Strobl and Hirzinger (2006) as well as Pradeep et al. (2014) use a checkerboard while Knoll (n.d); Yu and Xi (2018) as well as Wang et al. (2020) use one or multiple spheres. Kang et al. (2007) calibrated a robot with an attached line scanner by following a spanned string and in Rüther et al. (2010) the robot system detects markers at its joints by watching itself in a mirror. A wider overview of recent, traditional calibration approaches can be found in Li, Li, et al. (2021).

Just like for eye-to-hand calibration, there is a recent trend toward getting rid of calibration objects. While not being "calibration" in its traditional form, Klingensmith et al. (2016) presented a system to track the pose of an eye-in-hand depth sensor and feed the offset back into the manipulators control loop to compensate it. However, this method generates considerable, continuous computing payload and only works in environments providing sufficient surfaces for dense depth tracking. To address the later issue, Klingensmith (2016) proposed to add an landmarks-based localization using BRISK features from RGB-D data. The problem of markerless calibration on RGB-D was in fact solved by Li et al. (2019). However, their approach is based on AKAZE image features and thus only works for sensors providing 2D image data, whereas our approach only requires depth or range data and further supports other measurement geometries such as line scanners or single beam LiDARs.

5 | APPROACH

The aim of this work is to find the optimal parameters $\boldsymbol{k}_{\text{opt}}$ describing the kinematic model of a robot. In a first step multiple scans of the robot's environment are recorded by moving a depth sensor attached to the robot's EE. By exploiting the knowledge about the robot's design and configuration at the time of scanning, one can transform all acquired point cloud data to $\mathcal{B}$. In a static environment, all overlapping points from the performed scans must match to the same surfaces. Thus any errors in the fused reconstructions must originate from errors in the projection of measurements along the kinematic chain. By formulating a cost function to express the quality of matching points, one can use the ICP algorithm to minimize the projection error and thus estimate $\boldsymbol{k}_{\text{opt}}$.

5.1 | Data structure and notation

Let's consider a robot with an actuated depth sensor in an eye-in-hand configuration. As the used measuring technique and lens model may differ, for the scope of this work, the sensor is assumed to be intrinsically calibrated and capable of producing a set of points P. While it is negligible how the data was measured, it is essential to know when each depth value was obtained. Depending on the sensor type one obtains a different amount of data at time t. In detail, a single beam LiDAR only measures a single range value, a triangulation-based line scanner captures a vector of points, and a depth camera even returns an entire matrix of measurements. Thus, depending on the used sensor type the elements of P can be rearranged to form a matrix $\boldsymbol{P}$:

$$\boldsymbol{P}^{\text{LiDAR}} = \boldsymbol{p}, \tag{11}$$

$$\boldsymbol{P}^{\text{LineScanner}} = (\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_n)^{\mathsf{T}}, \tag{12}$$

and

$$\boldsymbol{P}^{\text{DepthCamera}} = \begin{bmatrix} \boldsymbol{p}_{1,1} & \boldsymbol{p}_{1,2} & \cdots & \boldsymbol{p}_{1,m} \\ \boldsymbol{p}_{2,1} & \boldsymbol{p}_{2,2} & \cdots & \boldsymbol{p}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{p}_{n,1} & \boldsymbol{p}_{n,2} & \cdots & \boldsymbol{p}_{n,m} \end{bmatrix}. \tag{13}$$

Recording multiple scans while moving the robot finally combines multiple $\boldsymbol{P}$ to a dataset tensor $\boldsymbol{D}$:

$$\boldsymbol{D}^{\text{LineScanner}} = \begin{pmatrix} \boldsymbol{P}_1^{\text{Line Scanner}^T} \\ \boldsymbol{P}_2^{\text{Line Scanner}^T} \\ \vdots \\ \boldsymbol{P}_m^{\text{Line Scanner}^T} \end{pmatrix}^{\mathsf{T}} \tag{14}$$

and

$$\boldsymbol{D}^{\text{DepthCamrea}} = \begin{pmatrix} \boldsymbol{P}_1^{\text{Depth Camrea}} \\ \boldsymbol{P}_2^{\text{Depth Camrea}} \\ \vdots \\ \boldsymbol{P}_m^{\text{Depth Camrea}} \end{pmatrix}. \tag{15}$$

Rotating LiDARs provide one special case, as they also measure a line but sequentially. To maintain the spatial order of points in a LiDAR scan, all l points from a single rotation are arranged column-wise, so that

$$\boldsymbol{D}^{\text{LiDAR}} = \begin{bmatrix} \boldsymbol{P}_{1,1}^{\text{LiDAR}} & \boldsymbol{P}_{1,2}^{\text{LiDAR}} & \cdots & \boldsymbol{P}_{1,m}^{\text{LiDAR}} \\ \boldsymbol{P}_{2,1}^{\text{LiDAR}} & \boldsymbol{P}_{2,2}^{\text{LiDAR}} & \cdots & \boldsymbol{P}_{2,m}^{\text{LiDAR}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{P}_{l,1}^{\text{LiDAR}} & \boldsymbol{P}_{l,2}^{\text{LiDAR}} & \cdots & \boldsymbol{P}_{l,m}^{\text{LiDAR}} \end{bmatrix}. \tag{16}$$

For each measurement $\boldsymbol{P}_t$ there is a matching vector with the robot's joint states $\boldsymbol{j}_t$. The issue of different measuring frequencies can be overcome via interpolation of the joint positions to find an approximation of the exact state for t. As a result there is a matching a matrix

$$\boldsymbol{J}_i = (\boldsymbol{j}_1, \boldsymbol{j}_2, ..., \boldsymbol{j}_m) \tag{17}$$

matching each $\boldsymbol{D}_i$, or a tensor

$$\boldsymbol{J}_i^{\text{LiDAR}} = \begin{bmatrix} \boldsymbol{j}_{1,1} & \boldsymbol{j}_{1,2} & \cdots & \boldsymbol{j}_{1,m} \\ \boldsymbol{j}_{2,1} & \boldsymbol{j}_{2,2} & \cdots & \boldsymbol{j}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{j}_{l,1} & \boldsymbol{j}_{l,2} & \cdots & \boldsymbol{j}_{l,m} \end{bmatrix}. \tag{18}$$

for each $\boldsymbol{D}_i^{\text{LiDAR}}$, respectively.

For the following formulations, the recorded data will be denoted as $\boldsymbol{D}$, regardless of the used sensor type. While the tensor $\boldsymbol{D}^{\text{DepthCamrea}}$ has a rank of three (four if one counts each point's x, y, z coordinates), I use the two-index notation $\boldsymbol{p}_{i,j}$ as an equivalent to

$$\boldsymbol{p}_{i,j} = \boldsymbol{p}_{i,j,k} \quad \forall\, k \tag{19}$$

for simplification. In the same way, we use $\boldsymbol{j}_{i,j}$ to address the joint states matching a point $\boldsymbol{p}_{i,j}$, even though $\boldsymbol{J}$ may have a different rank than $\boldsymbol{D}$.

5.2 | Parameter modeling

We use the MCPC model to express the spatial relationships of the robots kinematic chain. An initial guess $\boldsymbol{k}_{\text{init}}$ of $\boldsymbol{k}_{\text{opt}}$ can be obtained via manual measurements or from the robot's datasheet. Though the model MCPC itself is free of redundancies a few parameters remain which cannot be calibrated. As the transformation between $\mathcal{B}$ and the first joint of robot $\mathcal{J}_1$ behaves as a static offset to all recorded point clouds and there is no information to deduce its parameters. Also, as frame $\mathcal{J}_1$ is lacking a fixed position in space, two further redundancies arise in the parameters of $\boldsymbol{T}^{\mathcal{J}_2 \to \mathcal{J}_1}$. As such, y_0 and β_0 also have to be excluded from the optimization. We thus use a bitmask vector $\boldsymbol{m}$ of the same size as $\boldsymbol{k}$ with

$$\boldsymbol{m} = (m_1, m_2, ..., m_n) | m_i \in \{0, 1\} \quad \forall\, i \tag{20}$$

to define which parameters shall be taken into account in the optimization process. $\boldsymbol{m}$ is also used to reduce the number of DoFs for prismatic joints.

5.3 | Normalization

The influence of orientation errors on the cost metric heavily depends on the distance of the scanned surface to the sensor. To balance the weight of angular and translation parameters, $\boldsymbol{k}$ should be normalized before optimization. Since all data is captured by the robot scanning its environment, one can assume the data to be more or less equally distributed around the robot's base. Thus only approximating the scaling factor s from a subset of all datasets is usually sufficient. We project all points of the first recorded dataset $\boldsymbol{D}_1^{\mathcal{E}}$ to $\mathcal{B}$ to obtain a tensor of points $\boldsymbol{D}_1^{\mathcal{B}}$ with

$$\boldsymbol{p}_{i,j}^{\mathcal{B}} = \text{tr}(\boldsymbol{k}_{\text{init}}, \boldsymbol{j}_{i,j}) \boldsymbol{p}_{i,j}^{\mathcal{E}} \quad \forall\, \boldsymbol{p}_{i,j}^{\mathcal{B}} \in \boldsymbol{D}_1^{\mathcal{B}}, \boldsymbol{p}_{i,j}^{\mathcal{E}} \in \boldsymbol{D}_1^{\mathcal{E}}, \tag{21}$$

allowing to compute s by

$$s = \frac{1}{n} \sum_{i=1}^{n} \left\| \boldsymbol{p}_i^{\mathcal{B}} \right\| | \boldsymbol{p}_i^{\mathcal{B}} \in \boldsymbol{D}_1^{\mathcal{B}}. \tag{22}$$

We denote components normalized by s with a hat symbol (as in $\hat{\boldsymbol{p}}$). The normalization of a dataset $\boldsymbol{D}^{\mathcal{E}}$ is straightforward:

$$\hat{\boldsymbol{D}}^{\mathcal{E}} = \frac{1}{s} \boldsymbol{D}^{\mathcal{E}}. \tag{23}$$

Note that one also needs to normalize the translation parameters of $\boldsymbol{k}$

$$\hat{\boldsymbol{k}} = \left(\alpha_0, \beta_0, \frac{x_0}{s}, \frac{y_0}{s}, \alpha_1, \beta_1, \frac{x_1}{s}, \frac{y_1}{s}, \ldots, \alpha_n, \beta_n, \gamma, \frac{x_n}{s}, \frac{y_n}{s}, \frac{z}{s}\right)^{\mathrm{T}} \quad (24)$$

as well as the joint positions of prismatic joints.

Even though the parameters of $\boldsymbol{k}$ change with every ICP iteration, it is safe to keep s constant over the whole optimization process. As the accumulated error of $\boldsymbol{k}_{\text{init}}$ is usually in the range of a few centimeters, the numerical effects on s are negligible.

In case the initial assumption of an equal distribution of points around the robot's base does not hold, it might be necessary to (a) compute s by averaging points from all datasets, as well as (b) to add an additional translation $\boldsymbol{T}^{\mathcal{B}\to\mathcal{C}}$ shifting the points to their average centerpoint, which would need to be incorporated as an additional transformation in $\mathrm{tr}(\boldsymbol{k}, \boldsymbol{j})$ and masked in $\boldsymbol{m}$.

5.4 | ICP

The implemented ICP algorithm can be summarized in four steps (see below for a detailed explanation): (1) Initial projection and validation of points, (2) search for point matches, (3) verification of point matches, and (4) computation of the error function and optimization of $\hat{\boldsymbol{k}}$. All four steps are repeated iteratively until convergence is reached. A detailed overview of the used operations is given in Algorithm 1.

Algorithm 1 ICP for calibration

Require: Datasets $\left\{\hat{\boldsymbol{D}}_1^{\mathcal{E}}, \hat{\boldsymbol{D}}_2^{\mathcal{E}}, \ldots, \hat{\boldsymbol{D}}_n^{\mathcal{E}}\right\}$

Require: Joint positions $\left\{\hat{\boldsymbol{j}}_1^{\mathcal{E}}, \hat{\boldsymbol{j}}_2^{\mathcal{E}}, \ldots, \hat{\boldsymbol{j}}_n^{\mathcal{E}}\right\}$

Require: Initial guess $\boldsymbol{k}_{\text{init}}$ and parameter mask $\boldsymbol{m}$

Require: Stop threshold ϵ

$\hat{\boldsymbol{k}}_{\text{opt}} \leftarrow \text{normalizeTranslation}(\boldsymbol{k}_{\text{init}})$

repeat

Compute $\hat{\boldsymbol{D}}_1^{\mathcal{B}}, \hat{\boldsymbol{D}}_2^{\mathcal{B}}, \ldots, \hat{\boldsymbol{D}}_n^{\mathcal{B}}$: $\hat{\boldsymbol{p}}_t^{\mathcal{B}} = \mathrm{tr}(\hat{\boldsymbol{k}}_{\text{opt}}, \hat{\boldsymbol{j}}_t)\hat{\boldsymbol{p}}_t^{\mathcal{E}} \quad \forall\, t$

$\hat{\boldsymbol{O}}_i^{\mathcal{B}} \leftarrow \text{detectNoiseAndEdges}\left(\hat{\boldsymbol{D}}_i^{\mathcal{B}}\right) \quad \forall\, i$

$\hat{\boldsymbol{Q}}_i^{\mathcal{B}} \leftarrow \hat{\boldsymbol{D}}_i^{\mathcal{B}} \setminus \hat{\boldsymbol{O}}_i^{\mathcal{B}} \quad \forall\, i$

$M \leftarrow \{\}$

for each $\hat{\boldsymbol{Q}}_i^{\mathcal{B}}, \hat{\boldsymbol{Q}}_j^{\mathcal{B}} | i < j$ do

$M \cup \text{findMatches}\left(\hat{\boldsymbol{Q}}_i^{\mathcal{B}}, \hat{\boldsymbol{Q}}_j^{\mathcal{B}}\right)$

end for

$M \leftarrow \text{filter}(M)$

Solve $\min_{\hat{\boldsymbol{k}} | \hat{\boldsymbol{k}}_i \in \hat{\boldsymbol{k}}_{\text{opt}} \wedge m_i = 1 | m_i \in \boldsymbol{m}} \mathrm{e}(\hat{\boldsymbol{k}}_{\text{opt}}, M)$

$\boldsymbol{k}_{\text{opt}} \leftarrow \text{denormalizeTranslation}(\hat{\boldsymbol{k}}_{\text{opt}})$

until $\|\Delta \boldsymbol{k}_{\text{opt}}\| \leq \epsilon$

return $\boldsymbol{k}_{\text{opt}}$

Note that the projection of $\hat{\boldsymbol{D}}^{\mathcal{E}}$ to $\hat{\boldsymbol{D}}^{\mathcal{B}}$ is the result of a transformation along a kinematic chain, parameterized by joint states that change over the duration of the recording. In other words: Even though the individual transformations based on $\hat{\boldsymbol{k}}$ are themselves rigid, errors in the kinematic parameters will lead to nonlinear distortions as shown in Figure 2. This effect makes it hard to apply common tools often used in point cloud registration, as features, tree-structures for point matching and normals cannot be precomputed. This is a particularly limiting factor for the cost function, as numeric optimization of $\hat{\boldsymbol{k}}$ leads to an ongoing deformation of the resulting point cloud. We thus use the cross product for normal computation

$$\boldsymbol{n}_{i,j} = \mathrm{n}\left(\boldsymbol{p}_{i,j}^{\mathcal{B}}\right) = \frac{\mathrm{m}\left(\boldsymbol{p}_{i-1,j}^{\mathcal{B}}, \boldsymbol{p}_{i,j-1}^{\mathcal{B}}\right) + \mathrm{m}\left(\boldsymbol{p}_{i+1,j}^{\mathcal{B}}, \boldsymbol{p}_{i,j+1}^{\mathcal{B}}\right)}{\left\|\mathrm{m}\left(\boldsymbol{p}_{i-1,j}^{\mathcal{B}}, \boldsymbol{p}_{i,j-1}^{\mathcal{B}}\right) + \mathrm{m}\left(\boldsymbol{p}_{i+1,j}^{\mathcal{B}}, \boldsymbol{p}_{i,j+1}^{\mathcal{B}}\right)\right\|} \quad (25)$$

with

$$\mathrm{m}(\boldsymbol{p}, \boldsymbol{q}) = \frac{\boldsymbol{p} \times \boldsymbol{q}}{\|\boldsymbol{p} \times \boldsymbol{q}\|} \quad (26)$$

and the point-to-plane error metric as a compromise between runtime and precision. Since $\boldsymbol{n}_{i,j}$ is normalized, it is the same whether it is computed via $\mathrm{n}(\boldsymbol{p}_{i,j})$ or $\mathrm{n}(\hat{\boldsymbol{p}}_{i,j})$.

Note that for scans other than depth images, the normal orientation depends on the scanning trajectory and the resulting order of scan lines (e.g., whether the lines were recorded from left to right or from right to left). It may thus be necessary to verify the normal orientation by comparing it to the direction of the sensor's view ray:

$$\boldsymbol{v}_{i,j} = \mathrm{vn}\left(\boldsymbol{p}_{i,j}^{\mathcal{B}}\right) = \begin{cases} \mathrm{n}\left(\boldsymbol{p}_{i,j}^{\mathcal{B}}\right) & \mathrm{n}\left(\boldsymbol{p}_{i,j}^{\mathcal{B}}\right) \cdot \boldsymbol{o}_t^{\mathcal{B}} \leq 0 \\ -\mathrm{n}\left(\boldsymbol{p}_{i,j}^{\mathcal{B}}\right) & \text{else} \end{cases} \quad (27)$$

where the origin of the sensor $\boldsymbol{o}_t^{\mathcal{B}}$ at a time t is simply the translational part of $\boldsymbol{T}_t^{\mathcal{E}\to\mathcal{B}}$.

5.4.1 | Projection and validation

Before searching for point matches, all datasets $\hat{\boldsymbol{D}}_i^{\mathcal{E}}$ have to be projected to $\mathcal{B}$:

$$\hat{\boldsymbol{p}}_t^{\mathcal{B}} = \mathrm{tr}(\hat{\boldsymbol{k}}_{\text{init}}, \hat{\boldsymbol{j}}_t)\hat{\boldsymbol{p}}_t^{\mathcal{E}} \quad \forall\, \hat{\boldsymbol{p}}_t^{\mathcal{B}} \in \hat{\boldsymbol{D}}_i^{\mathcal{B}}, \hat{\boldsymbol{p}}_t^{\mathcal{E}} \in \hat{\boldsymbol{D}}_i^{\mathcal{E}} \quad \forall\, i. \quad (28)$$

It is natural for sensor data to contain noise and outliers, as well as, in the context of depth sensors, gaps. Moreover, the chosen approach for normal estimation is prone to making incorrect assumptions along corners and edges, an effect becoming especially prominent when the density between separate scan lines and points within a single scan line heavily differs (see Figure 3). To improve the

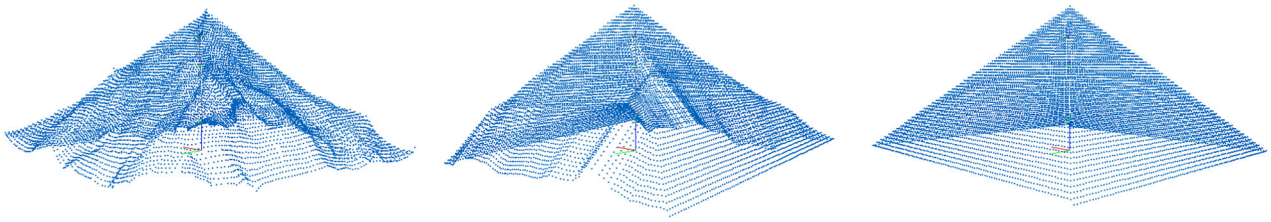

FIGURE 2 Distortion of point clouds obtained by projecting depth measurements of a pyramid along an imprecise kinematic model with randomized scanning trajectories. From left to right: One single point per joint configuration (single-beam LiDAR), one line per configuration (line scanner) and a matrix of points obtained from a single configuration (depth camera).

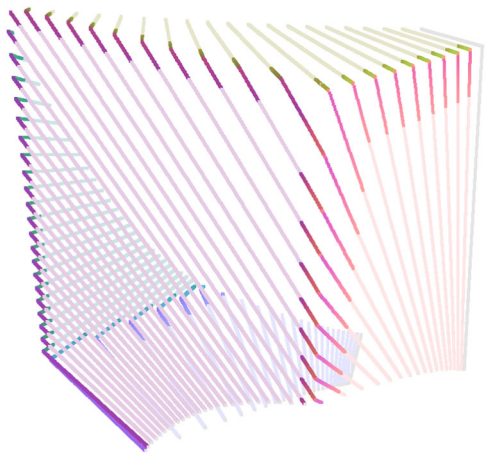

FIGURE 3 Scan lines of a cubic room, colorized by the direction of the points' normal vectors. For points along corners some of the neighboring points lie on other surfaces, causing the normals to bend toward edges. The effect becomes visible as a color gradient on the scan lines. Areas with uniform normals are rendered in lighter colors.

quality of the calibration result it is recommended to remove such measurements from the recorded data.

While the detection of invalid points (usually indicated by `NaN` values) is straightforward, the classification of outliers and edges requires an additional filtering routine. Unfortunately, many pre-processing filters, such as the one we initially used in Peters et al. (2020) do not generalize well over different sensor types as they make implicit or explicit assumptions regarding the sampling density. We thus apply a window of $n \times m$ radius each projected point $\hat{\boldsymbol{p}}_{i,j}^{\mathcal{B}}$ to compute overlap of neighboring normals

$$\mathrm{o}\left(\hat{\boldsymbol{p}}_{i,j}^{\mathcal{B}}\right) = \sum_{a=-n}^{n} \sum_{b=-m}^{m} \frac{\left\| \boldsymbol{v}_{i,j} \cdot \boldsymbol{v}_{i+a,j+b} \right\|}{(2n+1)(2m+1)} \tag{29}$$

and exclude all points with a value $\mathrm{o}(\hat{\boldsymbol{p}}_{i,j})$ below a threshold $g_{\min}$. By choosing different values for n and m it is possible to compensate for varying densities of points and scan lines. As the indices of points are the same in $\hat{\boldsymbol{D}}^{\mathcal{B}}$ and $\hat{\boldsymbol{D}}^{\mathcal{E}}$, one can also remove the respective measurements from the raw data. The filtered point clouds are denoted $\hat{\boldsymbol{Q}}^{\mathcal{B}}$ and $\hat{\boldsymbol{Q}}^{\mathcal{E}}$, respectively.

5.4.2 | Point matching

For each pair of filtered datasets $\langle \hat{\boldsymbol{Q}}_i^{\mathcal{B}}, \hat{\boldsymbol{Q}}_j^{\mathcal{B}} \rangle$ a k-d tree (Bentley, 1975) is constructed over $\hat{\boldsymbol{Q}}_j^{\mathcal{B}}$. It is used to find the closest neighbor $\hat{\boldsymbol{p}}_v^{\mathcal{B}} \in \hat{\boldsymbol{Q}}_j^{\mathcal{B}}$ for every point $\hat{\boldsymbol{p}}_u^{\mathcal{B}} \in \hat{\boldsymbol{Q}}_i^{\mathcal{B}}$. Based on the indices u and v a pair $\langle \hat{\boldsymbol{p}}_i^{\mathcal{E}}, \hat{\boldsymbol{p}}_j^{\mathcal{E}} \rangle$ is obtained and added to a set M.

Since—aside from the original ICP—bundle adjustment is used to align more than two scans at once, the process is repeated for all possible combinations of datasets (without respect to the order).

5.4.3 | Match validation

Unfortunately, pure nearest neighbor matching is prone to finding incorrect pairs, for example, when a point $\hat{\boldsymbol{q}}^{\mathcal{B}} \in \hat{\boldsymbol{Q}}_i^{\mathcal{B}}$ lies on a surface not captured in $\hat{\boldsymbol{Q}}_j^{\mathcal{B}}$. Thus all matches with a point-to-point distance

$$\mathrm{d}\left(\hat{\boldsymbol{q}}_i^{\mathcal{B}}, \hat{\boldsymbol{q}}_j^{\mathcal{B}}\right) = \left\| \hat{\boldsymbol{q}}_i^{\mathcal{B}} - \hat{\boldsymbol{q}}_j^{\mathcal{B}} \right\| \tag{30}$$

above a maximum distance $\hat{d}_{\max}$ and a normal overlap

$$\mathrm{f}\left(\hat{\boldsymbol{q}}_i^{\mathcal{B}}, \hat{\boldsymbol{q}}_j^{\mathcal{B}}\right) = \mathrm{vn}\left(\hat{\boldsymbol{q}}_i^{\mathcal{B}}\right) \cdot \mathrm{vn}\left(\hat{\boldsymbol{q}}_j^{\mathcal{B}}\right) \tag{31}$$

below a threshold $f_{\min}$ are excluded from M (similar to Newcombe et al., 2011).

5.4.4 | Cost function

The basic idea behind the point-to-plane distance is to measure the distances only perpendicular between a point and the target surface. This is commonly done by taking the dot product between the Cartesian distance vector and the surface's normal vector (Chen & Medioni, 1992). In our case the point-to-plane distance of a match $\langle \hat{\boldsymbol{q}}_i^{\mathcal{B}}, \hat{\boldsymbol{q}}_j^{\mathcal{B}} \rangle \in M$ is defined by

$$\mathrm{c}\left(\hat{\boldsymbol{q}}_i^{\mathcal{B}}, \boldsymbol{q}_j^{\mathcal{B}}, \hat{\boldsymbol{k}}\right) = \left[\mathrm{tr}(\hat{\boldsymbol{k}}, \hat{\boldsymbol{j}}_i) \hat{\boldsymbol{p}}_i^{\mathcal{E}} - \mathrm{tr}(\hat{\boldsymbol{k}}, \hat{\boldsymbol{j}}_j) \hat{\boldsymbol{p}}_j^{\mathcal{E}} \right] \cdot \mathrm{n}\left(\hat{\boldsymbol{q}}_i^{\mathcal{B}}\right) \tag{32}$$

resulting in a total error

$$\mathrm{e}(\hat{\boldsymbol{k}}, M) = \sum_{\left\langle \hat{\boldsymbol{q}}_i^{\mathcal{B}}, \hat{\boldsymbol{q}}_j^{\mathcal{B}} \right\rangle \in M} \mathrm{c}\left(\hat{\boldsymbol{q}}_i^{\mathcal{B}}, \hat{\boldsymbol{q}}_j^{\mathcal{B}}, \hat{\boldsymbol{k}}\right)^2. \tag{33}$$

Since the normals are only used as part of a squared dot product, one can safely ignore their sign in the computation of the error. The Levenberg-Marquard method (Levenberg, 1944; Marquardt, 1963) is used to find the optimal, unmasked parameters of $\hat{\boldsymbol{k}}_{\text{opt}}$ minimizing

$$\min_{\hat{\boldsymbol{k}} | \hat{k}_i \in \hat{\boldsymbol{k}}_{\text{opt}} \wedge m_i = 1 | m_i \in \boldsymbol{m}} \mathrm{e}(\hat{\boldsymbol{k}}_{\text{opt}}, M) \tag{34}$$

The final calibration parameters $\boldsymbol{k}_{\text{opt}}$ can then be obtained by removing the scaling factor s from $\hat{\boldsymbol{k}}_{\text{opt}}$:

$$\boldsymbol{k} = (\alpha_0, \beta_0, s\hat{x}_0, s\hat{y}_0, \alpha_1, \beta_1, s\hat{x}_1, s\hat{y}_1, \ldots, \alpha_n, \beta_n, \gamma, s\hat{x}_n, s\hat{y}_n, s\hat{z})^{\mathsf{T}}. \tag{35}$$

All four steps of the ICP algorithm described above are iteratively repeated until $\Delta\boldsymbol{k}_{\text{opt}}$ between two iterations reaches below a threshold ϵ.

6 | EVALUATION

The aforementioned formulations yield in a generic framework, theoretically enabling the calibration of any kinematic chain with an attached 3D sensor, as long as it is possible to obtain precise readings of the included joint positions at scan time. To demonstrate the capabilities of the proposed system, it is tested on a seven-DoF robot arm in combination with varying 3D sensors.

6.1 | Experimental setup

We used a KUKA LBR iiwa R840 manipulator (see Figure 4) and a total of four sensors with different characteristics: A Hokuyo UTM-30LX rotating single-beam LiDAR, a Wenglor MLSL236 triangulation based line scanner, a Microsoft Kinect Azure consumer grade depth camera and a PhotoNeo MotionCam 3D high end industrial grade depth camera (see Figure 5).

The LiDAR has by far the widest view range of all tested sensors, but is only meant for far range scanning. Moreover, the studies shown in Pomerleau et al. (2012) suggest that the sensor error in its measurement precision is similar to absolute Gaussian noise with a standard deviation of 1.8 cm.

In contrast the Wenglor MLSL236 is extremely precise, but its view range is limited to close range scenes. Unfortunately there are no studies about its sensor model available. However, the vendor's datasheet specifies the maximum depth error to stay below 600 µm (Wenglor Sensoric GmbH, 2020). Given the three-sigma-rule one thus may assume a Gaussian noise model with $\sigma_{\text{abs}} = 0.2$ mm.

Microsoft's Kinect Azure is the only consumer grade 3D sensor in the test field and with a recommended retail price of around 400 EUR by far the cheapest one. It is also the only sensor suitable for both close and far range scenes. The Kinect offers a *narrow field of view* (NFOV) as well as a *wide field of view* (WFOV) scanning mode, both which can be combined with an additional 2 × 2 binning, increasing the z-precision at a cost of the xy-resolution. All datasets of the Kinect Azure were captured in NFOV 2 × 2 binned mode. According to the findings of Tölgyessy et al. (2021) the sensor's noise model for this mode is similar to linear Gaussian noise modeled by

$$\sigma = \sigma_{\text{rel}} z + \sigma_{\text{abs}}, \tag{36}$$

with $\sigma_{\text{rel}} = 0.21\%$, $\sigma_{\text{abs}} = 2.53$ mm and z to be the distance of a single measurement.

Finally, the PhotoNeo MotionCam 3D is a high end industrial grade depth camera for scenes between 50 cm and 1 m distance. It can be seen as a successor to PhotoNeo's PhoXi 3D Scanner, adding support for dynamic scanning at up to 20 Hz. As the used sensor technology is very similar to the PhoXi 3D Scanner, one may take the observations presented in Cop and Peters (2021) as a point of reference for modeling the sensor noise: Averaged over four different materials σ_{abs} has a value of 0.18 mm.

An outline of the tested sensors' characteristics, as well as the used parameters is given in Table 1.

The evaluation is performed on two close range scenes of basic objects—a 3D printed Stanford Bunny (originally scanned and made public by Turk and Levoy using their back then new scanning

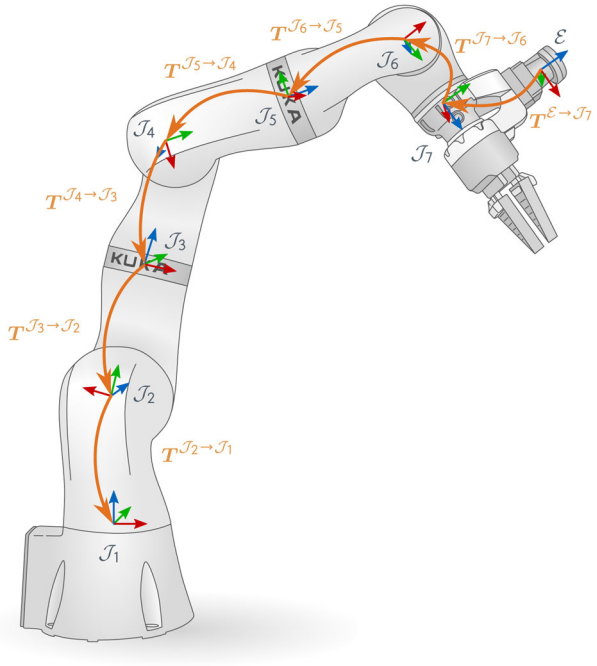

FIGURE 4 Physical joint frames of the KUKA LBR iiwa R840 robot. All joints rotate around their local z-axes. The base frame $\mathcal{B}$ of the kinematic chain is equivalent to the frame of the first joint $\mathcal{J}_1$. When applying the MCPC modeling convention, joints $\mathcal{J}_7$, $\mathcal{J}_5$, and $\mathcal{J}_3$ are shifted along their z-axes to the positions of their predecessors. Setting the non-calibratable parameters y and β of $\boldsymbol{T}^{\mathcal{J}_1 \to \mathcal{J}_2}$ to zero also shifts $\mathcal{J}_1$ (and thus also $\mathcal{B}$) up, thus positioning $\mathcal{J}_2$ in its xy-plane.

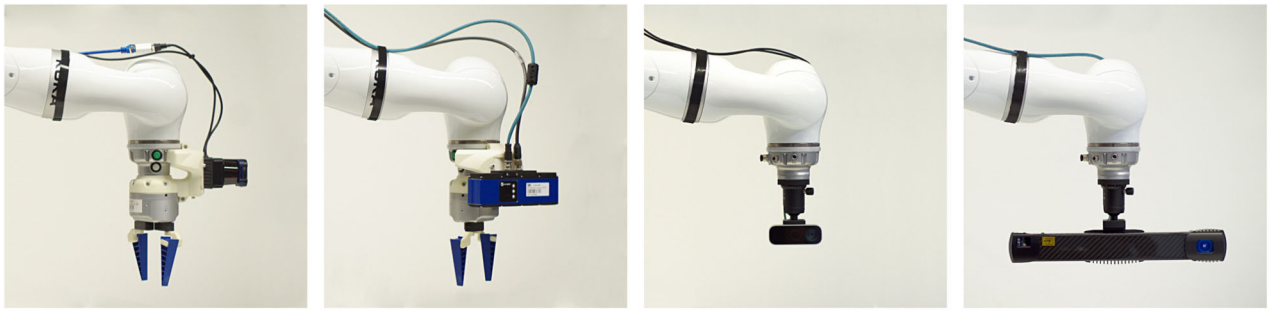

FIGURE 5 Used sensors from left to right: (1) Hokuyo UTM-30LX laser range finder, (2) Wenglor MLSL236 line scanner, (3) Microsoft Kinect Azure, and (4) PhotoNeo MotionCam 3D.

TABLE 1 Sensor characteristics and used parameters.

	Resolution			View range		Noise ratio		Preprocessing			Match validation		Stop criteria	
	x		y	Closest	Farthest	σ_{abs}	σ_{rel}	n	m	g_{min}	d_{max}	f_{min}	ϵ	i_{max}
Hokuyo UTM 30LX	1080	×	1[a]	10.0 cm	4000.0 cm	18.00 mm	–	2	4	0.60	20 mm	0.75	10^{-4}	50
Wenglor MLSL236	1280	×	1	30.0 cm	150.0 cm	0.20 mm	–	3	2	0.80	20 mm	0.80	10^{-4}	50
Microsoft Kinect Azure	320	×	288[b]	50.0 cm[d]	546.0 cm	2.53 mm	0.21%	2	2	0.75	20 mm	0.80	10^{-4}	50
PhotoNeo MotionCam 3D	1120	×	800[c]	49.7 cm	93.9 cm	0.18 mm	–	2	2	0.80	20 mm	0.80	10^{-4}	50

[a]All 1080 range values are recorded sequentially during rotation.

[b]NFOV 2 × 2 binned mode.

[c]Dynamic mode.

[d]According to datasheet. However, experiments have shown that the used objects were already measurable at less than 20 cm distance.

algorithm (Turk & Levoy, 1994)) and a Utah teapot (also known as Newell Teapot) (Crow, 1987; Newell, 1975) which is still in production by its original manufacturer—as well as two large scale scenes, featuring an office and a laboratory at TUM (see Figure 6). As the large scale scenes are the most challenging ones (see Section 6.3) we further use a virtual, cubic room of 10 m side lenght—similar to the use used by Oberländer et al. (2015) and Peters et al. (2020)—to test the convergence of our approach.

The Hokuyo LiDAR has an extensive reach and a view angle of 270° making its orientation inconsequential, as it is almost always capable of seeing a surface somewhere within most indoor scenes. Thus, recordings were performed by moving the robot to seven randomly generated configurations and moving one of it's axes by 180°, each at a time. The process was repeated five times, resulting in four batches with a total of 28 scans, along with one validation set with seven scans. The joint velocities were adjusted to obtain a similar density between scan lines and points within a single line.

For the other sensors the devices' fields of view must be taken into account when selecting the scanning poses to ensure sufficient overlap in the projected point clouds of the datasets. In these recordings, the scanning poses were chosen via manual selection of suitable sensor placements and the use of an inverse kinematics solver to find appropriate robot configurations. In the case of the Wenglor MLSL236, the trajectories were defined by a fixed start and end configuration in joint space. The robot was moved between those with a constant velocity to allow linear interpolation of the joints' positions. For the two depth cameras the robot was moved to fixed positions from which only single images were taken. Figure 7 provides an detailed overview of the selected trajectories. The same motions were also used for the generation of synthetic data. All used datasets are also available online at (Peters & Knoll, 2023).

Our runtime measurements were performed on a PC equipped with an AMD Ryzen 9 5950X CPU (2020 model with 16 physical cores) and 128 GB RAM, running a multithreaded C++ implementation of the described framework under Ubuntu 18.04 LTS. The maximum number of iterations i_{max} per test run was limited to 50.

6.2 | Reference calibration

A reference calibration was obtained using an optical tracking system based on five Vicon Vero v1.3 cameras. For this calibration a marker was placed on the ground next to the robot's base while a second one was mounted to the EE. The robot was then moved to 5000 randomly selected configurations while the poses of both markers as well as the robot's joint positions were recorded. 90% of those points were used to compute the optimal MCPC parameters connecting the static marker next to the robot's base $\mathcal{O}$ with the one attached to the EE $\mathcal{E}$, by minimizing

FIGURE 6 Pictures of the scanned real-world scenes. (1) Office, (2) TUM laboratory (the robot is positioned is in the workcell on the right), (3) Standford Bunny and (4) Utah Teapot (1.4 l version).

$$\min_{\boldsymbol{k}_{\text{traditional}}} \sum_i \text{cartesianDistance}(\boldsymbol{e}_i, \boldsymbol{m}_i)^2 + \text{angularDistance}(\boldsymbol{e}_i, \boldsymbol{m}_i)^2, \tag{37}$$

where $\boldsymbol{m}_i$ is ith the measured EE pose, $\boldsymbol{e}_i$ is the estimated one based on $\text{tr}(\boldsymbol{k}_{\text{tradional}}, \boldsymbol{j}_i)$ in meters, and $\text{angularDistance}(\boldsymbol{e}_i, \boldsymbol{m}_i)$ gives the smallest angle between both orientations in radians (as by in expressing the transformation in an angle-axis format). Based on the found model, outliers in the measurements with an offset of more than 5 cm and/or 0.05 rad (2.86°) from the estimated EE pose were excluded from the recorded data. We repeated the procedure for a total of five times.

The remaining 500 poses are used as validation dataset for both our approach as well as the reference calibration. Compared on this data, the traditional calibration reaches a positional error of 1.77 mm and an orientation error of 0.547° when applying the same outlier filtering on these measurements as well.

Note, that the kinematic chain calibrated through using the optical tracking system, starts with a static marker O placed next to the robot's base and ends with another marker attached to its EE $\mathcal{E}_{\text{Marker}}$, whereas our approach estimates the chain between the first joint $\mathcal{J}_1$ and the carried 3d sensor $\mathcal{E}_{\text{Sensor}}$. To enable an evaluation on the same 500 validation measurements, the parameters for the transformation between the seventh joint $\mathcal{J}_7$ and $\mathcal{E}$ in our solution are replaced with the ones obtained from the reference calibration. The same applies for the transformation from O and $\mathcal{J}_1$ as well as to the noncalibratable parameters between $\mathcal{J}_1$ and $\mathcal{J}_2$. As such, the differences in the precision originate solely from the estimated MCPC parameters of the robot itself. To ensure similar conditions, the outlier filtering on the verification data is applied here as well.

The initial model for both calibration attempts was obtained from CAD data of the used components whenever available. Parts for which such data was not accessible were measured manually. A total of four reference models were computed for comparison: (1) The uncalibrated model (URDF), (2) an uncalibrated robot at a known position, (3) an uncalibrated robot with a calibrated EE and (4) a fully calibrated system. An overview of the calibrated parameters by the different approaches is given in Table 2.

For tests of synthetic data, we sampled a total of 1500 poses using the known parameters of $\boldsymbol{k}$ and compared the pose of the EE, skipping the filtering of measurement outliers.

6.3 | Results

A distinct alignment of the obtained 3D data must be possible to reach convergence. While the exact scene or object itself does not matter, there should ideally be at least which normals span a space of rank three visible in each scan to prevent drift. It may be possible that

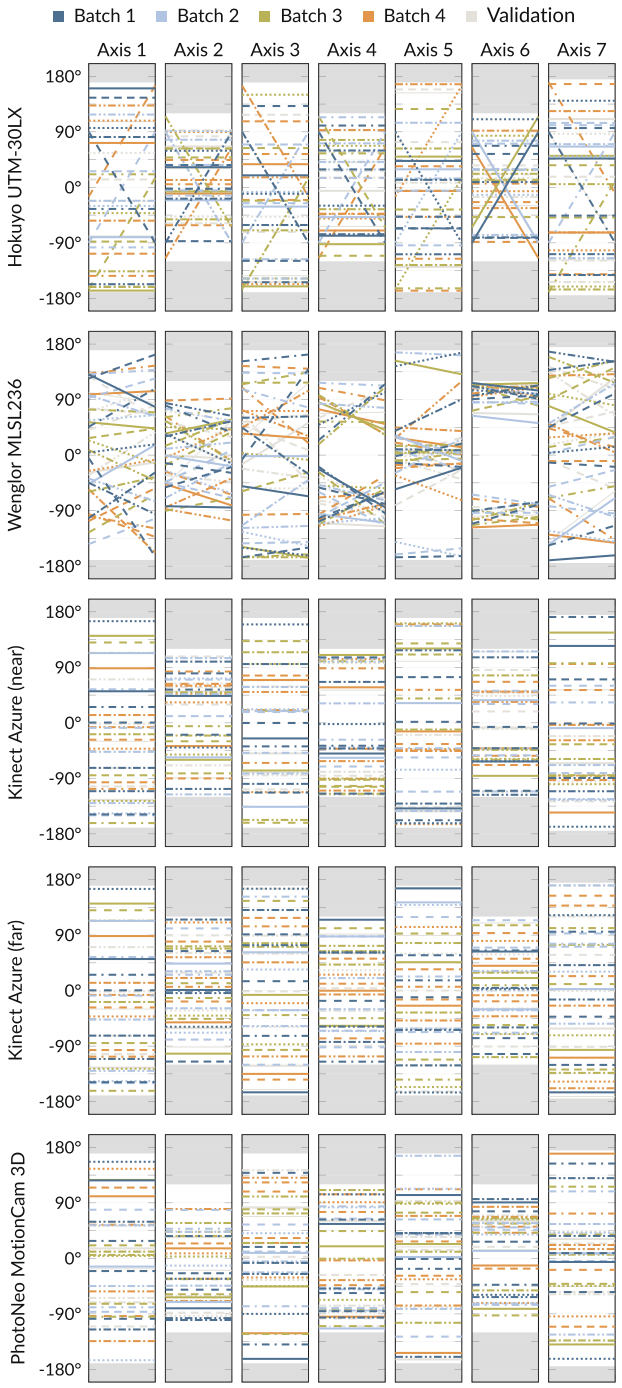

FIGURE 7 Scanning poses and trajectories used per sensor. Colors are defined by batch, while the line style identifies a single configuration. Areas outside of the joint axes motion ranges are grayed out. The scans of the first batch were used for the test runs with seven datasets. Datasets from the other batches were added subsequently to increase the overall number of datasets.

for certain kinematic chains and sufficiently large datasets there may even exist a global optima for simpler scenes. However, we did not investigate special case, as we assume it to be a purely theoretical scenario of limited practical use. The second requirement for our approach to work, is for the initial guess to be "well enough" to

TABLE 2 Overview of calibrated parameters.

	$\mathcal{O}$ to $\mathcal{J}_1$						$\mathcal{J}_1$ to $\mathcal{J}_2$					$\mathcal{J}_2$ to $\mathcal{J}_3$					$\mathcal{J}_3$ to $\mathcal{J}_4$					$\mathcal{J}_4$ to $\mathcal{J}_5$					$\mathcal{J}_5$ to $\mathcal{J}_6$					$\mathcal{J}_6$ to $\mathcal{J}_7$					$\mathcal{J}_7$ to $\mathcal{E}$					
	α	β	γ	x	y	z	α	β	x	y		α	β	x	y		α	β	x	y		α	β	x	y		α	β	x	y		α	β	γ	x	y		α	β	γ	x	y	z
Calibratable parameters	-	-	-	-	-	-	•	-	•	-		•	•	•	•		•	•	•	•		•	•	•	•		•	•	•	•		•	•	•	•	•		•	•	•	•	•	•
Model for comparison	◆	◆	◆	◆	◆	◆	•	◆	•	◆		•	•	•	•		•	•	•	•		•	•	•	•		•	•	•	•		•	•	•	•	•		◆	◆	◆	◆	◆	◆
URDF model	○	○	○	○	○	○	○	○	○	○		○	○	○	○		○	○	○	○		○	○	○	○		○	○	○	○		○	○	○	○	○		○	○	○	○	○	○
URDF + Calibrated origin	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆		○	○	○	○		○	○	○	○		○	○	○	○		○	○	○	○		○	○	○	○	○		○	○	○	○	○	○
URDF + Calibrated origin & EE	◆	◆	◆	◆	◆	◆	◆	○	◆	○		○	○	○	○		○	○	○	○		○	○	○	○		○	○	○	○		◆	◆	◆	◆	◆		◆	◆	◆	◆	◆	◆
Full reference calibration	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆		◆	◆	◆	◆		◆	◆	◆	◆		◆	◆	◆	◆		◆	◆	◆	◆		◆	◆	◆	◆	◆		◆	◆	◆	◆	◆	◆

Note: ○ Parameters from CAD model/manual measuring • Calibrated using presented approach ◆ Calibrated using optical tracking system.

provide at least some overlap of the visible surfaces, so that the numerical optimization is able to estimate a gradient for $e(\hat{\boldsymbol{k}}, M)$.

Figure 8 shows the convergence of 100 calibrations in a virtual, empty room. For the recordings we simulated a depth camera with the same parameters and Gaussian noise ratio as a Microsoft Kinect Azure. Whereas the room has a total of six corners, the cameras opening angle only allows to see one of them at a time. We used the same dataset and ground truth each scan, while the initial guess was randomized. In detail we introduced a total rotation offset of up to ±0.05 radians around a random axis and a translation of up ±5 cm in a random direction. This resulting initial mean EE pose offset of 0.069 rad (3.95°) and 77.9 mm is around twice as high as what we observed on real data.

As one can see, both overall performance is highly dependent on the amount of used depth recordings. Whereas the calibration did not reach convergence at all when only using seven recordings, we can already obtain reasonable calibration parameters from 14 scans. 98 of the 100 runs reached the desired ϵ in less than 60 ICP iterations. Subsequent tests have shown that the other two runs are

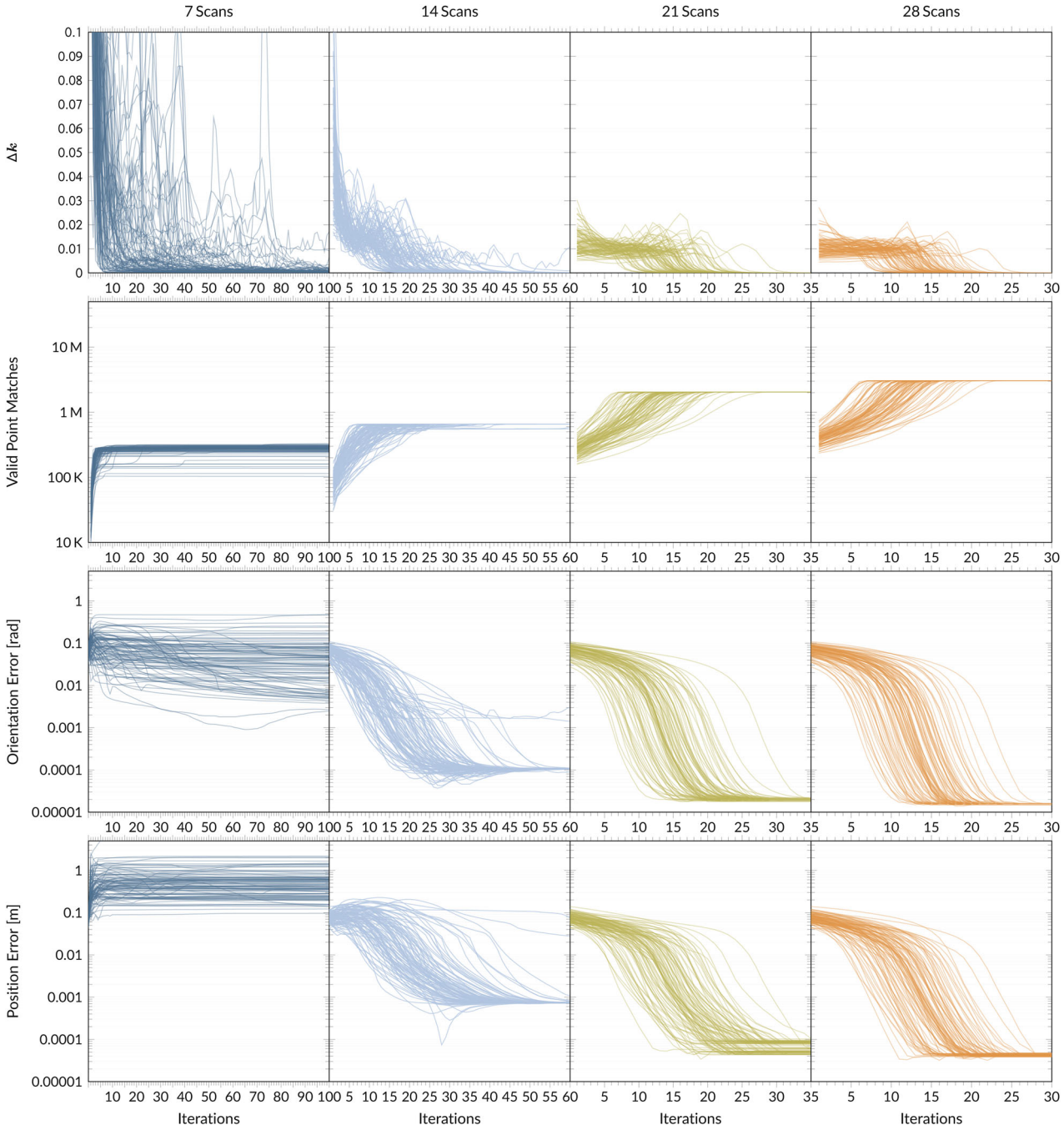

FIGURE 8 Development of the $\Delta\boldsymbol{k}$, the number of total point matches as well as the orientation and position error over multiple iterations, while running a calibration on different numbers of scans of a virtual, cubic room. Iteration zero shows the errors in the initial guess. Figure derived from Peters (2023).

TABLE 3 Comparison of our results to traditional calibration on a verfifacation dataset obtained by using a 3D tracking system.

	Dataset	Scans	Orientation error (°)	Position error (mm)	Outliers	Filtered orientation error (°)	Filtered position error (mm)	Iterations	Valid point matches per iteration	Computation runtime (s)
Hokuyo UTM-30LX	Lab	7	1.287	25.71	25	0.744	22.95	50	1,190,814.7	5573.0
		14	1.178	10.45	23	0.636	7.39	16	5,411,943.3	8129.9
		21	1.148	10.68	23	0.604	7.59	8	11,413,095.6	11,076.0
		28	1.114	11.76	23	0.567	8.77	7	21,016,877.9	18,932.5
	Office	7	1.217	18.80	24	0.675	15.87	50	1,970,957.2	9997.9
		14	1.103	7.94	23	0.556	4.80	9	9,062,888.7	8835.3
		21	1.112	7.65	23	0.565	4.48	14	19,224,321.9	25,507.3
		28	1.115	7.25	23	0.569	4.06	7	35,045,811.3	27,312.3
Wenglor MLSL236	Bunny	7	1.267	7.62	23	0.727	4.45	50	1,092,781.1	6627.7
		14	1.102	5.08	23	0.555	1.80	17	5,979,392.7	9014.5
		21	1.108	5.04	23	0.562	1.76	7	12,093,808.4	9182.5
		28	1.108	5.15	23	0.561	1.87	7	21,902,965.6	16,269.6
	Teapot	7	1.138	5.77	23	0.592	2.51	50	1,408,625.3	7617.3
		14	1.100	5.47	23	0.553	2.19	7	7,820,361.6	5564.8
		21	1.116	5.32	23	0.569	2.04	8	17,363,749.6	13,996.8
		28	1.116	5.24	23	0.571	1.96	6	33,530,538.7	21414.3
Microsoft Kinect Azure	Lab	7	1.299	12.73	24	0.757	9.84	12	140,739.9	97.8
		14	1.111	6.74	23	0.564	3.55	12	239,020.8	186.1
		21	1.104	7.27	23	0.556	4.07	12	382,920.5	333.1
		28	1.101	6.79	23	0.552	3.57	16	724,856.7	819.4
	Office	7	1.364	59.93	329	0.822	35.13	18	249,166.5	317.2
		14	1.103	7.33	23	0.555	4.13	8	540,193.1	324.2
		21	1.108	6.35	23	0.561	3.10	8	1,408,679.1	642.3
		28	1.102	5.78	23	0.553	2.50	7	2,133,716.0	684.7
	Bunny	7	1.566	27.33	24	1.037	14.42	16	576,383.5	444.4
		14	1.172	10.48	23	0.626	7.43	10	2,452,292.9	1256.3
		21	1.116	7.15	23	0.567	3.97	8	5,009,795.9	2135.6
		28	1.114	6.50	23	0.566	3.28	9	8,736,878.1	4226.9
	Teapot	7	1.311	10.17	24	0.770	7.03	8	590,497.3	257.4
		14	1.142	7.72	23	0.594	4.57	8	2,465,504.3	1082.6
		21	1.103	5.91	23	0.555	2.67	7	5,184,669.3	2065.8
		28	1.103	5.34	23	0.556	2.07	6	9,190,931.3	3228.6
PhotoNeo MotionCam 3D	Bunny	7	1.190	18.48	25	0.642	15.60	38	1,666,129.5	3639.1
		14	1.109	5.48	23	0.562	2.21	18	6,206,851.7	6034.9
		21	1.101	5.11	23	0.553	1.82	14	14,566,960.8	11,531.8
		28	1.094	4.99	23	0.546	1.70	11	22,572,019.6	15,984.8
	Teapot	7	1.166	11.31	23	0.622	8.27	47	1,511,654.7	3951.3
		14	1.140	5.68	23	0.595	2.41	35	4,468,177.3	8495.6
		21	1.108	5.25	23	0.561	1.97	18	10,240,472.9	10,479.5
		28	1.096	5.10	23	0.548	1.82	15	14,713,290.9	12,979.9
Reference calibration	URDF	–	2.057	35.90	73	1.522	35.32			
	+ Origin	4500	2.016	24.46	27	1.484	21.91			
	+ EE	4500	1.103	6.24	23	0.556	3.00			
	Full	4500	1.095	5.05	23	0.547	1.77			

Note: The plain errors are the means EE offsets computed against all poses in the verification dataset. In the filtered errors, measurement outliers of the tracking system with more than 5 cm and/or 0.05 rad (2.86°) from the estimated EE poses were discarded from the verification data.

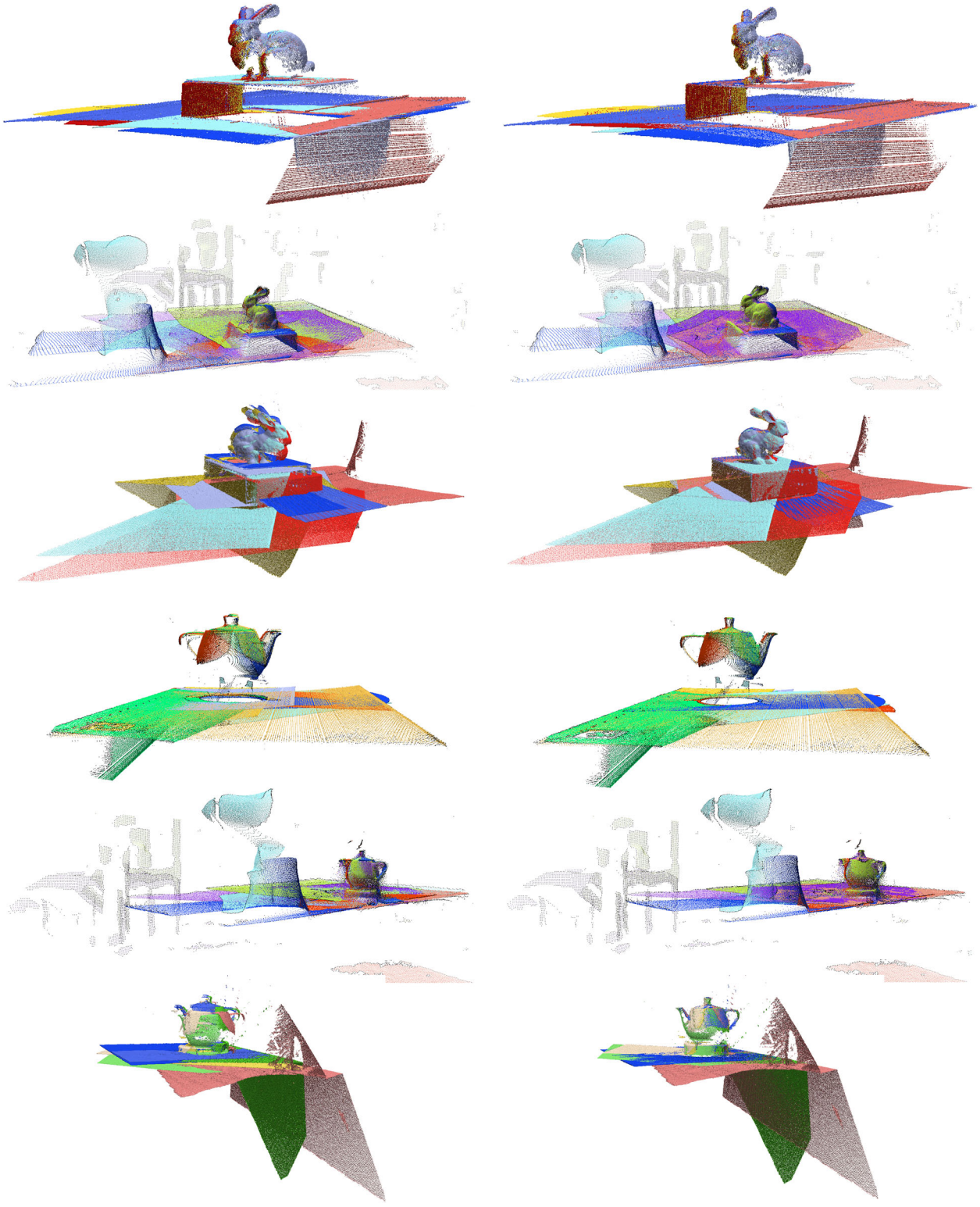

FIGURE 9 Point clouds projected by uncalibrated (left) and calibrated systems (right) on 28 datasets. Each pictures shows seven validation datasets that were not used for the calibration itself. From top to bottom: Stanford Bunny recorded with (1) Wenglor MLSL236, (2) Kinect Azure, and (3) Photoneo MotionCam, followed by Utah Teapot recorded with (4) Wenglor MLSL236, (5) Kinect Azure, and (6) Photoneo MotionCam. Note that the wooden warmer below the Teapot is not visible to the UV laser of the Wenglor scanner. Differences in the alignment of the point clouds of the Kinect Azure are best noticeable on the partly visible background structures.

about to converge after 88 and 96 iterations, respectively. Again, increasing the number of datasets to 21 or 28 leads to both faster convergence and even more precise results.

We can further observe spikes in $\Delta \boldsymbol{k}$ in many runs, shortly before actual convergence is reached. This effect is most likely caused by the used match validation strategy: If the overall orientation error is too big, the distance between points lying in the same plane can become quite big and there are only very few valid matches in areas where the surfaces intersect. However, once the error becomes small enough for the point match distances to reach below $d_{\max}$, the overall number of point matches drastically increases, causing the projected point clouds to "snap" together.

Given the simplicity of the test scene, we conclude that the calibration precision is mostly determined by the coverage of the robot's joint space, rather than the scanned room or object. The same observations can also be made in the calibration results on real-world data listed in Table 3.

As one can see, using only seven scans is usually insufficient to reach a suitable calibration result. Either the precision is far worse than what can be produced via the use of a larger number of scans, or the ICP did not converge at all. As expected, there is also a clear trend towards reaching a higher precision when using more data.

Also, the findings show a direct relation between the calibration result and the precision of the used sensor. For high-precision sensors such as the Wenglor MLSL236 or the PhotoNeo MotionCam 3D the results are similar to the ones obtained by the tracking system. In the case of the MotionCam 3D on the Stanford Bunny scene, the proposed framework even found a solution which is slightly better than the reference. However, even with a Kinect Azure one may obtain results almost as good a as the reference, regardless of its cost being less than one-50th of the tracking system's one. Figures 9 and 10 further show the projected points clouds of the validation datasets to allow a subjective impression of the achieved calibration quality.

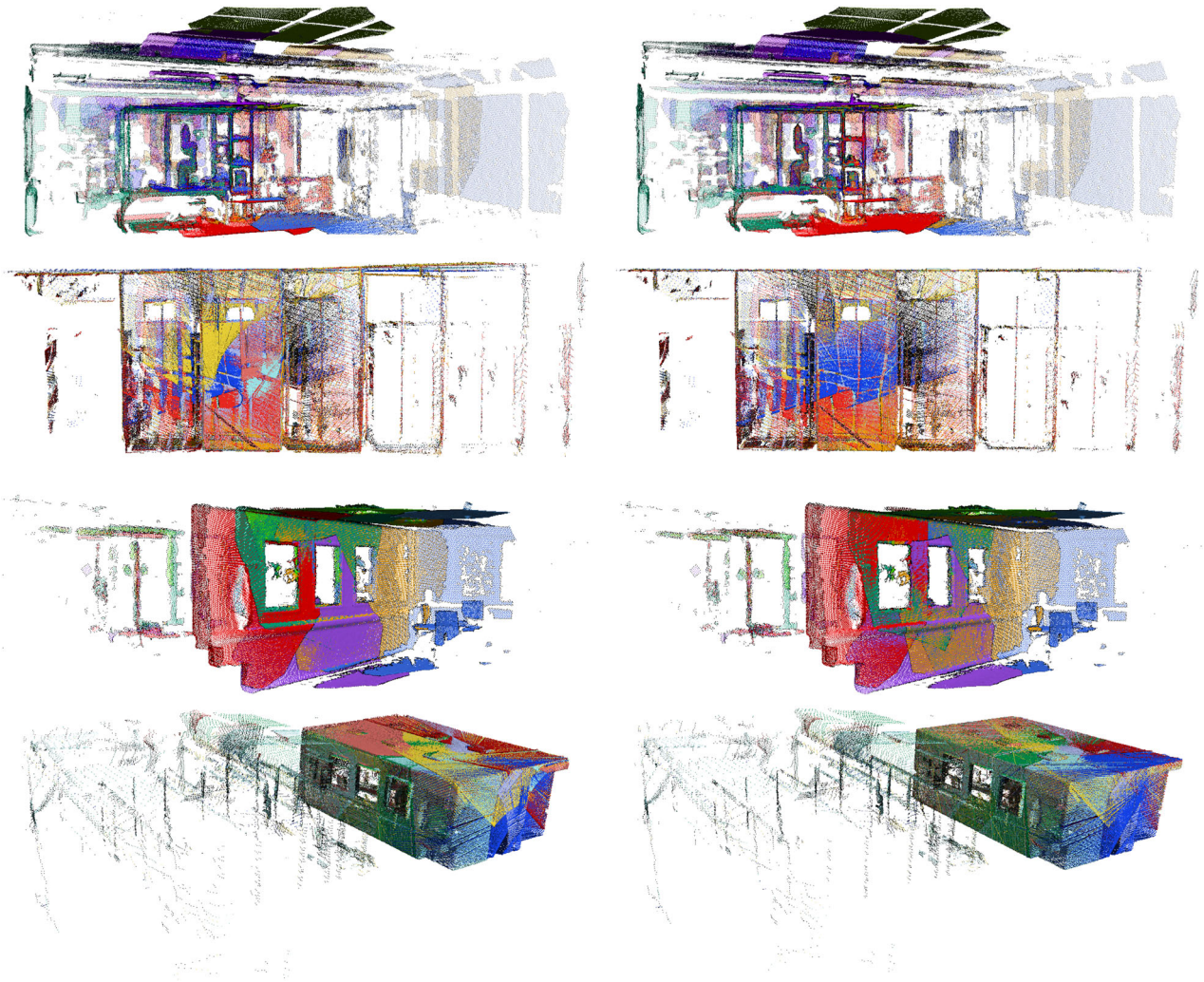

FIGURE 10 Point clouds projected by uncalibrated (left) and calibrated systems (right) on 28 datasets. Each pictures shows seven validation datasets that were not used for the calibration itself. From top to bottom: TUM laboratory recorded with (1) Kinect Azure, and (2) Hokuyo UTM-30LX (viewed from top), followed by an office recorded with (3) Kinect Azure, and (4) Hokuyo UTM-30LX.

The observations further suggest that less complex, small-scale scenes are better suited for calibration. Not only is the expected measurement error smaller for close distances, but especially in combination of large scenes captured by far range sensors even smallest orientation errors have a strong effect on the overall error metric. On the example of the lab scan taken with the Hokuyo LiDAR one can observe that the orientation error is continuously reduced with an increasing number of used datasets, even at the expense of the position error.

Evidently, the runtime is directly dependent on the number of used iterations and found point matches—which is again related to the number of scans. Fortunately, one can see that the number of required ICP iterations goes down when the overall number of datasets increases.

Finally, when recording the used datasets, we did not notice any impact of the Cartesian sensor poses on the calibration result. As such, we conclude, that the spatial position of the sensor is—with the exception of ensuring sufficient overlap in the scan data—negligible. It is, however, desirable to reach a complete and uniform sampling of the manipulator's joint space. Especially extrapolations in joint ranges not covered by the calibration data will result in errors. When closely studying Figure 7 one can see a direct relation between the coverage of the joint space and the calibration precision.

7 | SUMMARY

By extending the ICP algorithm to allow optimization of complex kinematic models instead of estimating a single, rigid transformation, a new framework for fully autonomous calibration of robot manipulators has been presented. The shown implementation has been evaluated on multiple real world scenes along with various hardware configurations. Comparing the results to a dedicated tracking system has clearly demonstrated the functionality of the framework. More than that: The achieved precision is comparable to the far more expensive reference system. Given the right scene, even a Microsoft Kinect Azure consumer grade depth camera can achieve a precision that is only few tens of millimeters off.

Having shown that self-calibration of robotic system is possible, multiple possible extensions to the formulated approach remain to be investigated: As the system is already capable of undistorting scans obtained from projections along a badly parameterized kinematic chain, it should also be possible to include a sensor's intrinsic parameters in the optimization. Additionally, finding formulations to deal with less precise actuator readings, such as odometry, would allow for further expansion of possible use cases. Finally, many strategies for optimizing the runtime of the ICP algorithm have been demonstrated. It is not unlikely that some of those are applicable in the context of this calibration problem as well.

ACKNOWLEDGMENTS

This article is part of a project that has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 870133. Special thanks goes to Daniel Hettegger and Bare Luka Žagar for their continuous assistance in the lab and their remarks to the contents and structure of this work. This also applies to Dinesh Paudel who was a great help in the assembly of the used robot workcell and to Michael Zechmair for his detailed feedback on the draft of this article. The presented C++ implementation would not have been possible without many contributors of open source libraries, most importantly Eigen (Guennebaud & Jacob, 2010) and Ceres Solver (Agarwal et al., 2023). Also great thanks to Salvatore Virga for publishing his iiwa_stack (Hennersperger et al., 2017). Open Access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Zenodo at https://zenodo.org/record/8310089, reference number https://doi.org/10.5281/zenodo.8310089.

ORCID

Arne Peters https://orcid.org/0000-0002-0620-3154
Alois C. Knoll https://orcid.org/0000-0003-4840-076X

REFERENCES

Abderrahim, M., Khamis, A., Garrido, S. & Moreno, L. (2004) Accuracy and calibration issues of industrial manipulators. In: Huat, L. K. (Ed.) *Industrial robotics: programming, simulation and application*. Rijeka: IntechOpen, pp. 131–146.

Agarwal, S., Mierle, K. & The Ceres Solver Team. (2023) Ceres solver. http://ceres-solver.org

Alismail, H., Baker, L.D. & Browning, B. (2012) Automatic calibration of a range sensor and camera system. In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE, pp. 286–292.

Alismail, H. & Browning, B. (2015) Automatic calibration of spinning actuated lidar internal parameters. *Journal of Field Robotics*, 32, 723–747.

Amberg, B., Romdhani, S. & Vetter, T. (2007) Optimal step nonrigid ICP algorithms for surface registration. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1–8.

American Automobile Association, Inc. (2018) *New vehicle technologies double repair bills for minor collisions*. Available from: https://newsroom.aaa.com/2018/10/new-vehicle-technologies-double-repair-bills-minor-collisions/

Andersen, T.T., Andersen, N.A. & Ravn, O. (2014) Calibration between a laser range scanner and an industrial robot manipulator. In: *2014 IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*. IEEE, pp. 1–8.

Andreff, N., Horaud, R. & Espiau, B. (1999) On-line hand-eye calibration. In: *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No. PR00062)*. IEEE, pp. 430–436.

Antone, M. & Friedman, Y. (2007) Fully automated laser range calibration. In: *BMVC 2007—Proceedings of the British Machine Vision Conference* 2007.

Atanacio-Jiménez, G., González-Barbosa, J.-J., Hurtado-Ramos, J.B., Ornelas-Rodríguez, F.J., Jiménez-Hernández, H., García-Ramirez, T., et al. (2011) Lidar velodyne hdl-64e calibration using pattern planes. *International Journal of Advanced Robotic Systems*, 8, 59.

Attia, M. & Slama, Y. (2017) Efficient initial guess determination based on 3d point cloud projection for icp algorithms. In: *2017 International Conference on High Performance Computing Simulation (HPCS)*, pp. 807–814.

Bennett, D. & Hollerbach, J. (1991) Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints. *IEEE Transactions on Robotics and Automation*, 7, 597–606.

Bentley, J.L. (1975) Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18, 509–517.

Besl, P.J. & McKay, N.D. (1992) Method for registration of 3-d shapes. In: *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611, International Society for Optics and Photonics, pp. 586–606.

Bi, C., Fang, J., Li, K. & Guo, Z. (2017) Extrinsic calibration of a laser displacement sensor in a non-contact coordinate measuring machine. *Chinese Journal of Aeronautics*, 30, 1528–1537. https://www.sciencedirect.com/science/article/pii/S1000936117301255

Birbach, O., Frese, U. & Bäuml, B. (2015) Rapid calibration of a multi-sensorial humanoid's upper body: an automatic and self-contained approach. *The International Journal of Robotics Research*, 34, 420–436.

Brown, B.J. & Rusinkiewicz, S. (2007) Global non-rigid alignment of 3-d scans. In: *ACM SIGGRAPH 2007 papers*, pp. 21–es.

Carlson, F.B., Johansson, R. & Robertsson, A. (2015) Six DoF eye-to-hand calibration from 2d measurements using planar constraints. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3628–3632.

Chen, S., Liu, J., Wu, T., Huang, W., Liu, K., Yin, D., et al. (2018) Extrinsic calibration of 2D laser rangefinders based on a mobile sphere. *Remote Sensing*, 10, 1176.

Chen, Y. & Medioni, G. (1992) Object modelling by registration of multiple range images. *Image and Vision Computing*, 10, 145–155.

Chen-Gang, Li-Tong, Chu-Ming, Xuan, J.-Q. & Xu, S.-H. (2014) Review on kinematics calibration technology of serial robots. *International Journal of Precision Engineering and Manufacturing*, 15, 1759–1774.

Cheng, S., Marras, I., Zafeiriou, S. & Pantic, M. (2017) Statistical non-rigid icp algorithm and its application to 3d face alignment. *Image and Vision Computing*, 58, 3–12.

Cop, K.P., Borges, P.V. & Dubé, R. (2018) Delight: an efficient descriptor for global localisation using lidar intensities. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3653–3660.

Cop, K.P., Peters, A., Zagar, B.L., Hettegger, D. & Knoll, A.C. (2021) New metrics for industrial depth sensors evaluation for precise robotic applications. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (ed. IEEE).

Crow, F. (1987) The origins of the teapot. *IEEE Computer Graphics and Applications*, 7, 8–19.

Cyganek, B. & Siebert, J.P. (2011) *An introduction to 3D computer vision techniques and algorithms*. Chichester: John Wiley & Sons.

Denavit, J. & Hartenberg, R.S. (1955) A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 215–221.

Diez, Y., Roure, F., Lladó, X. & Salvi, J. (2015) A qualitative review on 3d coarse registration methods. *ACM Computing Surveys (CSUR)*, 47, 1–36.

Fuchs, S. & Hirzinger, G. (2008) Extrinsic and depth calibration of tof-cameras. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1–6.

Guennebaud, G. & Jacob, B. (2010) Eigen v3. http://Eigen.tuxfamily.org

Hansard, M., Lee, S., Choi, O. & Horaud, R.P. (2012) *Time-of-flight cameras: principles, methods and applications*. Springer Science & Business Media.

Hartley, R.I. (1997) In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 580–593.

Heide, N., Emter, T. & Petereit, J. (2018) Calibration of multiple 3d lidar sensors to a common vehicle frame. In: *ISR 2018; 50th International Symposium on Robotics. VDE*, pp. 1–8.

Heller, J., Havlena, M. & Pajdla, T. (2015) Globally optimal hand-eye calibration using branch-and-bound. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38, 1027–1033.

Hennersperger, C., Fuerst, B., Virga, S., Zettinig, O., Frisch, B., Neff, T., et al. (2017) Towards mri-based autonomous robotic us acquisitions: a first feasibility study. *IEEE Transactions on Medical Imaging*, 36, 538–548.

Jang, J.H., Kim, S.H. & Kwak, Y.K. (2001) Calibration of geometric and non-geometric errors of an industrial robot. *Robotica*, 19, 311–321.

Judd, R. & Knasinski, A. (1990) A technique to calibrate industrial robots with experimental verification. In: *IEEE Transactions on Robotics and Automation*. vol. 6, pp. 20–30.

Kang, H.-J., Jeong, J.-W., Shin, S.-W., Suh, Y.-S. & Ro, Y.-S. (2007) Autonomous kinematic calibration of the robot manipulator with a linear laser-vision sensor. In: *International Conference on Intelligent Computing*. Springer, pp. 1102–1109.

Kannala, J. & Brandt, S.S. (2006) A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28, 1335–1340.

Klingensmith, M. (2016) *Automatically tracking and calibrating robot arms using SLAM techniques*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Klingensmith, M., Sirinivasa, S.S. & Kaess, M. (2016) Articulated robot motion for simultaneous localization and mapping (arm-slam). *IEEE Robotics and Automation Letters*, 1, 1156–1163.

Knoll, A.C. (n.d) Einrichtung und verfahren zum vermessen von mechanismen und ihrer stellung.

Levenberg, K. (1944) A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2, 164–168.

Li, J., Ito, A., Yaguchi, H. & Maeda, Y. (2019) Simultaneous kinematic calibration, localization, and mapping (skclam) for industrial robot manipulators. *Advanced Robotics*, 33, 1225–1234.

Li, J., Zhu, J., Guo, Y., Lin, X., Duan, K., Wang, Y., et al. (2008) Calibration of a portable laser 3-d scanner used by a robot and its use in measurement. *Optical Engineering*, 47, 017202.

Li, M., Du, Z., Ma, X., Dong, W. & Gao, Y. (2021) A robot hand-eye calibration method of line laser sensor based on 3D reconstruction. *Robotics and Computer-Integrated Manufacturing*, 71, 102136.

Li, Z., Li, S. & Luo, X. (2021) An overview of calibration technology of industrial robots. *IEEE/CAA Journal of Automatica Sinica*, 8, 23–36.

Lightcap, C., Hamner, S., Schmitz, T. & Banks, S. (2008) Improved positioning accuracy of the pa10-6ce robot with geometric and flexibility calibration. *IEEE Transactions on Robotics*, 24, 452–456.

Longuet-Higgins, H.C. (1981) A computer algorithm for reconstructing a scene from two projections. *Nature*, 293, 133–135.

Ma, Y., Soatto, S., Kosecka, J. & Sastry, S.S. (2012) *An invitation to 3-d vision: from images to geometric models*, vol. 26. Springer Science & Business Media.

Marquardt, D.W. (1963) An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11, 431–441.

Maye, J., Sommer, H., Agamennoni, G., Siegwart, R. & Furgale, P. (2016) Online self-calibration for robotic systems. *The International Journal of Robotics Research*, 35, 357–380.

Miseikis, J., Glette, K., Elle, O.J. & Torresen, J. (2016) Automatic calibration of a robot manipulator and multi 3d camera system. In: *2016 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, pp. 735–741.

Muhammad, N. & Lacroix, S. (2010) Calibration of a rotating multi-beam lidar. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5648–5653.

Mustafa, S.K., Yang, G., Yeo, S.H. & Lin, W. (2008) Kinematic calibration of a 7-dof self-calibrated modular cable-driven robotic arm. In: *2008 IEEE International Conference on Robotics and Automation*, pp. 1288–1293.

Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., et al. (2011) Kinectfusion: real-time dense surface mapping and tracking. In: *ISMAR*, vol. 11, pp. 127–136.

Newell, M.E. (1975) *The utilization of procedure models in digital image synthesis*. Technical report, Utah University Salt Lake City School of Computing.

Oberländer, J., Pfotzer, L., Roennau, A. & Dillmann, R. (2015) Fast calibration of rotating and swivelling 3-d laser scanners exploiting measurement redundancies. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3038–3044.

Özgüner, O., Shkurti, T., Huang, S., Hao, R., Jackson, R.C., Newman, W.S., et al. (2020) Camera-robot calibration for the da vinci robotic surgery system. *IEEE Transactions on Automation Science and Engineering*, 17, 2154–2161.

Peters, A. (2023) *Autonomous robot calibration using actuated 3D sensors*. Ph.D. thesis, Technical University of Munich, Munich, Germany. Under review.

Peters, A. & Knoll, A.C. (2023) *Datasets for article "robot self-calibration using actuated 3d sensors"*. Available from: https://zenodo.org/record/8310089

Peters, A., Schmidt, A. & Knoll, A.C. (2020) Extrinsic calibration of an eye-in-hand 2d lidar sensor in unstructured environments using ICP. *IEEE Robotics and Automation Letters*, 5, 929–936.

Pomerleau, F., Breitenmoser, A., Liu, M., Colas, F. & Siegwart, R. (2012) Noise characterization of depth sensors for surface inspections. In: *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*. IEEE, pp. 16–21.

Pomerleau, F., Colas, F. & Siegwart, R. (2015) A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4, 1–104.

Pradeep, V., Konolige, K. & Berger, E. (2014) Calibrating a multi-arm multi-sensor robot: a bundle adjustment approach. In: *Experimental Robotics*. Springer, pp. 211–225.

Richardson, A., Strom, J. & Olson, E. (2013) AprilCal: assisted and repeatable camera calibration. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Rusinkiewicz, S. (2019) A symmetric objective function for icp. *ACM Transactions on Graphics (TOG)*, 38, 1–7.

Rusinkiewicz, S. & Levoy, M. (2001) Efficient variants of the ICP algorithm. In: *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, pp. 145–152.

Rüther, M., Lenz, M. & Bischof, H. (2010) The narcissistic robot: robot calibration using a mirror. In: *2010 11th International Conference on Control Automation Robotics & Vision*. IEEE, pp. 169–174.

Segal, A., Haehnel, D. & Thrun, S. (2009) Generalized-ICP. In: *Robotics: science and systems*, vol. 2, Seattle, WA, p. 435.

Sheehan, M., Harrison, A. & Newman, P. (2014) Automatic self-calibration of a full field-of-view 3d n-laser scanner. In: *Experimental Robotics*. Springer, pp. 165–178.

Stone, H.W. (1987) *Kinematic modeling, identification, and control of robotic manipulators*, vol. 29. Springer Science & Business Media.

Strobl, K.H. & Hirzinger, G. (2006) Optimal hand-eye calibration. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4647–4653.

Sturm, P. & Ramalingam, S. (2011) *Camera models and fundamental concepts used in geometric computer vision*. Now Publishers Inc.

Tölgyessy, M., Dekan, M., Chovanec, L. & Hubinskỳ, P. (2021) Evaluation of the azure kinect and its comparison to kinect v1 and kinect v2. *Sensors*, 21, 413.

Tsai, R.Y. & Lenz, R.K. (1989a) Overview of a unified calibration trio for robot eye, eye-to-hand, and hand calibration using 3D machine vision. In: P.S. Schenker (Ed.) *Sensor fusion: Spatial reasoning and scene interpretation*, vol. 1003, International Society for Optics and Photonics, SPIE, pp. 202–213. https://doi.org/10.1117/12.948932

Tsai, R.Y. & Lenz, R.K. (1989b) A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5, 345–358.

Turk, G. & Levoy, M. (1994) Zippered polygon meshes from range images. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pp. 311–318.

Wagner, M., Heß, P., Reitelshöfer, S. & Franke, J. (2015) Self-calibration method for a robotic based 3d scanning system. In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2015, October.

Wang, J., Shi, F., Zhang, J. & Liu, Y. (2008) A new calibration model of camera lens distortion. *Pattern Recognition*, 41, 607–615.

Wang, R., Wu, A., Chen, X. & Wang, J. (2020) A point and distance constraint based 6r robot calibration method through machine vision. *Robotics and Computer-Integrated Manufacturing*, 65, 101959.

Wang, Y. & Solomon, J.M. (2019a) Deep closest point: learning representations for point cloud registration. *IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, pp. 3522–3531. https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00362

Wang, Y. & Solomon, J.M. (2019b) PRNet: self-supervised learning for partial-to-partial registration. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Garnett, R. (Eds.) *Advances in Neural Information Processing Systems*. Curran Associates, Inc., vol. 32, pp. 8814–8826. https://proceedings.neurips.cc/paper_files/paper/2019/file/ebad33b3c9fa1d10327bb55f9e79e2f3-Paper.pdf

Wenglor Sensoric GmbH. (2020) *2D/3D profile sensor: MLSL236*.

Xu, J., Hoo, J.L., Dritsas, S. & Fernandez, J.G. (2022) Hand-eye calibration for 2d laser profile scanners using straight edges of common objects. *Robotics and Computer-Integrated Manufacturing*, 73, 102221. https://www.sciencedirect.com/science/article/pii/S0736584521001046

Yamazoe, H., Habe, H., Mitsugami, I. & Yagi, Y. (2012) Easy depth sensor calibration. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, pp. 465–468.

Yin, S., Ren, Y., Guo, Y., Zhu, J., Yang, S. & Ye, S. (2014) Development and calibration of an integrated 3d scanning system for high-accuracy large-scale metrology. *Measurement*, 54, 65–76. https://www.sciencedirect.com/science/article/pii/S0263224114001675

Yu, C. & Xi, J. (2018) Simultaneous and on-line calibration of a robot-based inspecting system. *Robotics and Computer-Integrated Manufacturing*, 49, 349–360.

Yu, H., Li, F., Saleh, M., Busam, B. & Ilic, S. (2021) CoFiNet: reliable coarse-to-fine correspondences for robust pointcloud registration. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S. & Wortman Vaughan, J. (Eds.) *Advances in Neural Information Processing Systems*. Curran Associates, Inc., vol. 34, pp. 23872–23884. https://proceedings.neurips.cc/paper_files/paper/2021/file/c85b2ea9a678e74fdc8bafe5d0707c31-Paper.pdf

Zhang, S. (2018) High-speed 3d shape measurement with structured light methods: a review. *Optics and Lasers in Engineering*, 106, 119–131.

Zhang, Z. (2000) A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 1330–1334.

Zhang, Z., Zhang, L. & Yang, G.-Z. (2017) A computationally efficient method for hand-eye calibration. *International Journal of Computer Assisted Radiology and Surgery*, 12, 1775–1787.

Zhuang, H., Roth, Z.S. & Hamano, F. (1992) A complete and parametrically continuous kinematic model for robot manipulators. In: *IEEE Transactions on Robotics and Automation*, vol. 8, IEEE, pp. 451–463.
Zhuang, H., Wang, L.K. & Roth, Z.S. (1993) Error-model-based robot calibration using a modified CPC model. *Robotics and Computer-integrated Manufacturing*, 10, 287–299.

How to cite this article: Peters, A. & Knoll, A. C. (2024) Robot self-calibration using actuated 3D sensors. *Journal of Field Robotics*, 41, 327–346. https://doi.org/10.1002/rob.22259