Depth Sensor Calibration by Tracking an Extended Object

Florian Faion, Marcus Baum, Antonio Zea, and Uwe D. Hanebeck

***Abstract*— In this paper, we propose a novel algorithm for automatically calibrating a network of depth sensors, based on a moving calibration object. The sensors may have non-overlapping fields of view in order to avoid interference. Two major challenges are discussed. First, depending on where the object is located relative to the sensor, the number and quality of the measurements strongly varies. Second, a single depth sensor observes the calibration object only from one side. Dealing with these challenges requires a simple calibration object as well as an algorithm that can deal with under-determined measurements of varying quality. A recursive Bayesian estimator is developed that determines the extrinsic parameters by measuring the surface of a moving cube with known pose. Our approach does not restrict the configuration of the network and requires no manual initialization or interaction. Ambiguities that are induced by the rotational cube symmetries are resolved by applying a multiple model approach. Besides synthetic evaluation we perform real data experiments and compare to state-of-the-art calibration.**

I. Introduction

Depth sensors such as laser scanners or time-of-flight cameras produce noisy 3D point clouds of an environment. Networks of depth sensors became highly relevant to many fields, including robotics, surveillance, innovative control for entertainment devices, and telepresence applications. With the launch of Microsoft Kinect in 2010, a new class of low-cost depth sensors based on structured light has entered the scene, marking a significant technological leap. However, when two depth sensors share the same field of view, interference is an inherent problem of the active measurement principle, used by those sensors. In consequence, one has to deal with sensor scheduling techniques, e.g., time multiplexing [1], [2], or avoid overlapping fields of view. In this work, we consider the latter case, when sensor are mounted statically with a minimum of overlap.

One essential task in every network is the extrinsic calibration of the sensors, i.e., the calculation of the relative rigid transformations between the individual sensors. However, standard approaches cannot be applied in the non-overlap scenario, as they need shared features that are visible from at least two sensors. Another challenge is, that the measured points are distorted by sensor-specific noise, which is reflected by the number of measured points, as well as the degree of their uncertainty. In addition, due to the network configuration, occlusion and sensor artifacts can occur. For an accurate calibration, these challenges have to be taken into account.

All authors are with Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Germany. florian.faion@kit.edu

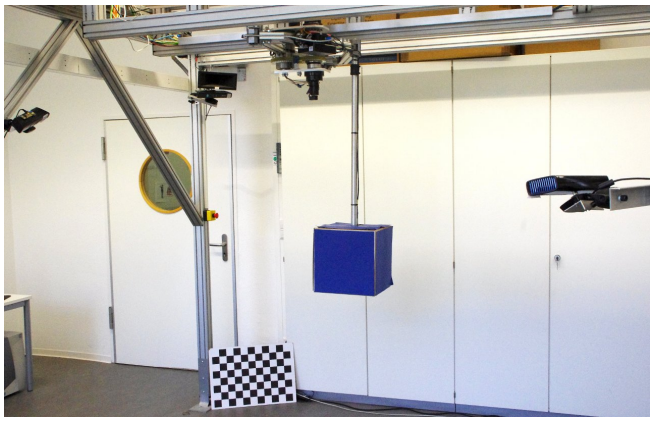


Fig. 1: Test setup with three Kinect depth sensors and a *blue* cube that is attached at a positioning unit. This unit provides the pose data of the cube.

A. Related Work

Even though extrinsic calibration of depth sensors is a well-studied problem, it is still in focus of recent research. Approaches can roughly be grouped into those that extract features from a known (1D, 2D, 3D) calibration object and the ones that extract features from the entire, potentially unknown scene. As calibration objects can serve checkerboards [3], [2], [4], planes [5], artificial markers [6], and other patterns [7], [8]. Simple 3D objects, as spheres [9] and cubes [10], [11] were also used for calibration. In [12], the practical relevance of a calibration method based on a known moving cube is explicitly shown. An example of estimating extrinsic parameters based on *iterative closest point* (ICP) for an entire scene is given in [13]. In previous work [14], we proposed a method to recursively calibrate rigidly linked depth sensors with non-overlapping fields of view. However, this approach requires all sensors to perform a common motion that is not practicable in our scenario. Simultaneous shape estimation and tracking of 3D objects based on point cloud measurements [15] is also related to this work.

B. Contribution

The main contribution of this work is a novel approach to calibrate a network of depth sensors by tracking the camera pose relative to a moving object. Because we cannot make any assumptions on shared fields of view, the global object poses have to be known in advance. This can be realized manually by mounting the object on a tripod and placing it at distinct positions or by means of a positioning unit such as used in [16], [12]. From a practical point of view, the proposed calibration approach comes with

1) automatic initialization, and
2) no restrictions on the configuration of the network, i.e., no overlapping views are required.

We evaluated the algorithm in an experimental environment that consists of a positioning unit [16] that moves a cube calibration object and three Kinect devices. The setup is shown in Fig. 1.

C. Discussion

The motivation for using a cube object for calibration is due to the following constraints [17]. On the one hand, the calibration object should be as simple as possible to allow for processing measurements of low quality. A detailed asymmetrical 3D object cannot be accurately estimated, based on a worst case of one measurement at a time step. Another reason for symmetric objects is their simpler segmentation, as they look the same from different directions. On the other hand, the calibration object should be as complex as possible to allow for distinct informative measurements. A sphere, or a cylinder for example, are invariant to rotations around a certain axis, which results in infinite symmetric ambiguities.

In sum, we chose a cube, which is as an object with 24 discrete rotational symmetries. A Cuboid object with three different edge lengths would be also reasonable, as it has only eight discrete rotational symmetries. However, simulations showed that estimation convergence is better by using equal edge lengths, i.e., a cube. In addition, and for the purpose of compatibility to depth-only sensors, color information for associating measurements to individual faces is not desired.

D. Overview

The remainder of the paper is structured as follows. In Sec. II, we give a mathematical problem formulation and an outline to the key idea of our approach. All required formulas for the Bayesian estimator are derived in Sec. III. The resulting calibration algorithm is explained in Sec. IV. In Sec. V, an analysis of the performed experiments with synthetic and real data is discussed. The paper concludes with a summary and an outlook to future work in Sec. VI.

II. Problem Formulation

In this work, we consider the extrinsic calibration of depth sensors based on noisy point cloud measurements of a known moving 3D object. We focus on a cube object that is assumed to have a known constant edge length l^c and, at a time step k, the pose

$$\mathbf{H}_k^c = \begin{bmatrix} \mathbf{R}_k^c & \underline{t}_k^c \\ \underline{0}^\mathrm{T} & 1 \end{bmatrix} \quad . \tag{1}$$

$\mathbf{H}_k^c$ is a homogeneous transformation matrix, i.e., $\underline{t}_k^c \in \mathbb{R}^3$ represents the position of the cube and $\mathbf{R}_k \in \mathrm{SO}(3)$ its orientation. At each time step k, a set of n 3D points $\hat{\mathcal{Y}}_k^s = \{\hat{\underline{y}}_{i,k}^s \mid i = 1 \dots n\}$ from the surface is measured by depth sensor s. These measurements are given in the local coordinate system of the depth sensor. In order to consider the varying measurement quality, each measurement[1] $\hat{\underline{y}}^s \in \hat{\mathcal{Y}}^s$ is assumed to be affected by additive Gaussian noise $\underline{\boldsymbol{w}}^s \sim \mathcal{N}(\underline{0}, \mathbf{C}_{\hat{\underline{y}}^s})$, so that

$$\hat{\underline{y}}^s = \underline{y}^s + \underline{\boldsymbol{w}}^s \quad . \tag{2}$$

Note that within this paper, we assume that the individual measurements $\hat{\underline{y}}^s \in \hat{\mathcal{Y}}^s$ are mutually independent. In addition, due to self-occlusion effects, the depth sensor observes the cube from one side at a given time step. Extrinsic calibration of the depth sensor s means estimating its pose relative to the global coordinate system. Analogously to the cube, this pose is represented as a homogeneous transformation matrix

$$\mathbf{H}^s = \begin{bmatrix} \mathbf{R}^s & \underline{t}^s \\ \underline{0}^\mathrm{T} & 1 \end{bmatrix} \quad . \tag{3}$$

A visualization of the parameters introduced in the problem formulation is shown in Fig. 2. In the following, a method is derived that recursively estimates $\mathbf{H}^s$ based on sequentially given point measurements $\hat{\mathcal{Y}}_k^s$, and cube poses $\mathbf{H}_k^c$.

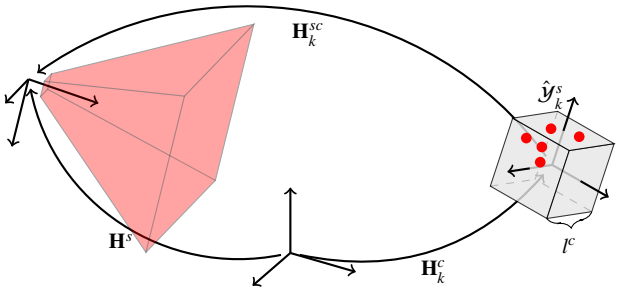


Fig. 2: Visualization of the problem formulation. The pose $\mathbf{H}^s$ of the depth sensor is to be estimated, based on sets of noisy surface measurements $\hat{\mathcal{Y}}_k^s$ of a moving cube with known, constant edge length l^c and poses $\mathbf{H}_k^c$. The measurements are given in local coordinates of the depth sensor coordinate system.

A. Key Idea

Essentially, the given task of finding the sensor poses $\mathbf{H}^s$ is identical to estimating their relative poses

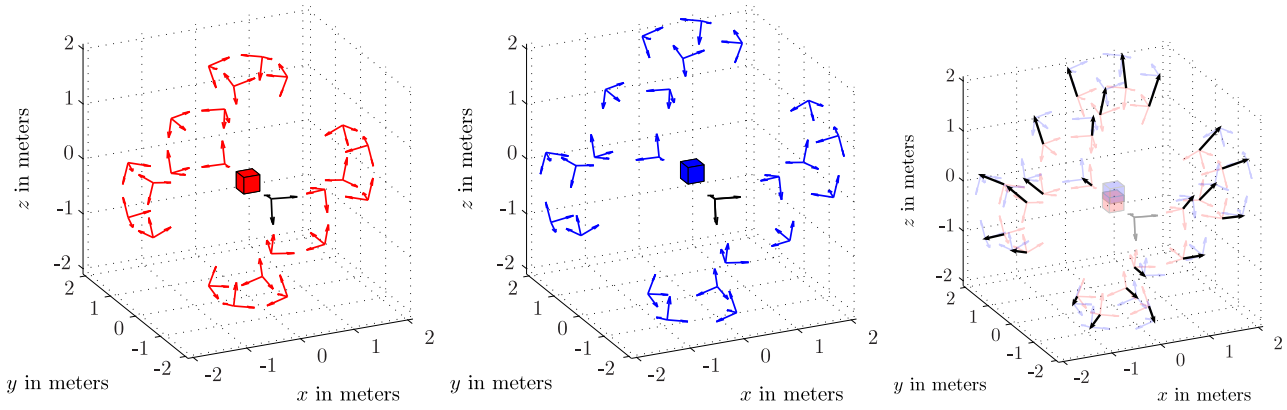
$$\mathbf{H}_k^{sc} = (\mathbf{H}_k^c)^{-1}\mathbf{H}^s \tag{4}$$

to the cube. This relation is shown in Fig. 2. In this work, we propose a recursive Bayesian state estimator for $\mathbf{H}^s$. This approach will allow for explicitly incorporating the measurement quality by means of a sensor model. In addition, the derived measurement equation will allow for updating the extrinsic parameters $\mathbf{H}^s$ with partial measurements, by separately processing each individual point of the cloud.

III. Recursive Bayesian Estimation

A Bayesian state estimator can be employed to recursively estimate an unknown parameter vector $\underline{x}$ based on underdetermined noisy information. Within the Bayesian framework, the estimation of this state parameter is modeled as a random vector $\underline{\boldsymbol{x}}_k$, where k denotes a specific time step. The recursion then consists of two steps. First, the *prediction step*

[1] Note that if possible, time indices k and measurement indices i have been omitted for clarity.



(a) Cube at location A. (b) Cube at location B. (c) Both cubes overlayed.

Fig. 3: Visualization of the 24 rotational cube symmetries for a cube at two diffrent locations A and B, seen from the same depth sensor *black*. The ambiguous sensor poses are drawn in *red* and *blue*, respectively. In (c), both examples are overlaid. While the true depth sensor remains at its initial pose, the symmetric sensors move as the cube moves.

lets the state $\underline{\boldsymbol{x}}_k$ evolve to the next time step $k+1$ according to a system model. Second, the *measurement update step* incorporates new measurements $\hat{\mathcal{Y}}_{k+1}$ according to Bayes' Rule.

A. State Representation

According to (3), the desired extrinsic parameters are the entries of the homogeneous transformation matrix $\mathbf{H}^s$. As the rotation matrix $\mathbf{R}^s$ has nine mutually dependent, partially redundant parameters, this representation is unsuitable for estimation. Following [14], we use the axis-angle representation as a 3×1 rotation vector $\underline{u}^s$ instead. Thus, the extrinsic parameters can be written as a 6×1 vector

$$\underline{x}^s = \begin{bmatrix} \underline{u}^s \\ \underline{t}^s \end{bmatrix} \quad . \tag{5}$$

Based on this representation, the state of the depth sensor is modeled as a Gaussian random vector $\underline{\boldsymbol{x}}_k^s \sim \mathcal{N}(\underline{\hat{x}}_k^s, \mathbf{C}_{\underline{x}_k}^s)$.

B. Time Update

All depth sensors s are assumed to be rigidly mounted. Thus, the system behavior is modeled as approximately static

$$\underline{x}_{k+1}^s = \underline{x}_k^s + \underline{\boldsymbol{v}}_k \quad , \tag{6}$$

where the system noise is assumed to be zero-mean Gaussian with $\underline{\boldsymbol{v}}_k \sim \mathcal{N}(\underline{0}, \mathbf{C}_{\underline{v}})$.

C. Measurement Update

In order to design a measurement equation, the relation between state $\underline{x}^s$ and measurements $\hat{\mathcal{Y}}^s$ has to be specified. For a more convenient notation of the cube pose $\mathbf{H}^c$, we also use rotation vectors instead of the matrix entries, yielding the stacked cube parameters

$$\underline{c}_k = \begin{bmatrix} \underline{u}_k^c \\ \underline{t}_k^c \\ l^c \end{bmatrix} \quad ,$$

where $\underline{u}_k^c$ represents the orientation, $\underline{t}_k^c$ the position, and l^c the constant edge length.

Object Model: Let us now specify the relationship between a depth sensor pose $\underline{x}^s$ and a measurement $\hat{\underline{y}}^s \in \hat{\mathcal{Y}}^s$ from a given cube $\underline{c}$ in local depth sensor coordinates. For this purpose, let the function $\mathrm{g} : \mathbb{R}^3 \times \mathbb{R}^7 \to \mathbb{R}$ implement the Euclidean distance from 3D points $\underline{y} \in \mathbb{R}^3$ (in global coordinates) to a cube given by its parameters $\underline{c} \in \mathbb{R}^7$. In particular, for points $\underline{y} \in \mathbb{R}^3$ on the cube $\underline{c}$

$$\mathrm{g}(\underline{y}, \underline{c}) = 0 \tag{7}$$

holds. Note that applying $\mathrm{g}(\cdot,\cdot)$ to 3D points $\underline{y}^s$ in local coordinates of a depth sensor s requires the transformation[2] of $\underline{y}^s$ to the global coordinate system according to

$$\underline{y} = \mathbf{H}^s \cdot \underline{y}^s \quad . \tag{8}$$

Measurement Equation: A measurement equation [18] that incorporates the sensor-specific measurement noise can be specified by plugging (2) into (7) according to

$$\mathrm{g}(\mathbf{H}^s \cdot (\hat{\underline{y}}^s - \underline{\boldsymbol{w}}^s), \underline{c}) = 0 \quad . \tag{9}$$

This nonlinear forward model can be used to perform a measurement update by means of a sampling-based nonlinear Kalman filters, e.g., an Unscented Kalman Filter (UKF) [19].

D. Sensor Model

We use the sensor model described in [1] to calculate the Kinect-specific uncertainty $\mathbf{C}_{\hat{\underline{y}}^s}$ for each 3D point $\hat{\underline{y}}^s$.

E. Symmetry

A cube, also referred to as a *regular hexahedron*, has 24 rotational symmetries. In consequence, there are 24 perspectives, where the cube looks equal to the sensor. Fig. 3a-b illustrates this issue for a cube at two different poses. The true sensor pose is indicated by the *black* arrows,

[2]For brevity, matrix multiplication between a 4×4 transformation matrix and a 3×1 vector is overloaded to additionally apply homogenization and de-homogenization.

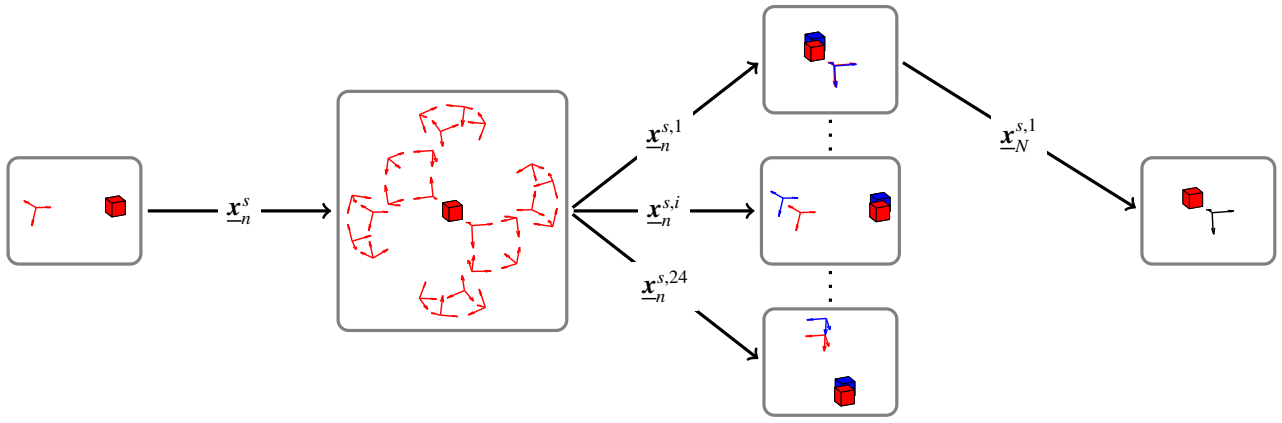


Fig. 4: Scheme of the proposed estimation algorithm. After the automatic initialization step, symmetric ambiguities are calculated and resolved by a multiple model approach. Finally, the desired result is refined.

while its ambiguous versions are depicted in *red* and *blue*, repsectively. The ambiguous state parameters will be denoted as $\underline{x}^{s,1}, \ldots, \underline{x}^{s,24}$, respectively $\mathbf{H}^{s,1}, \ldots, \mathbf{H}^{s,24}$. Their explicit derivation is given in Appendix A and essentially is a rotational transformation around the cube.

Remark 1 (Symmetry and uni-modal Estimators) *Due to the* 24 *rotational symmetries of a cube, a pose estimate* $\underline{\boldsymbol{x}}_k^s$ *of an uni-modal filter will converge to one of the ambiguous states* $\underline{x}_k^{s,1}, \ldots, \underline{x}_k^{s,24}$*, depending on the initialization of* $\underline{\boldsymbol{x}}_0^s$*. Note that in the presence of high measurement noise, the estimator could even converge to a local minimum between two poses. However, we assume the cube size to be chosen such that an uni-modal estimator will always find one of the valid poses.*

However, the pose ambiguities of the depth sensors can be resolved by incorporating a change of the cube pose. As stated in Sec. II, the depth sensors are rigidly mounted. In consequence, the true extrinsic parameters will converge to static values, even if the cube changes its pose. In contrast, the ambiguous pose parameters will not converge to static values as the cube moves. This relationship is visually explained in Fig. 3c by overlaying the examples from Fig. 3(a-b). Note that the true *black* coordinate system from (a) and (b) is perfectly overlaid. The ambiguous *red* and *blue* coordinate systems differ, as they are calculated by rotating the *black* depth sensor around the different cube locations. In the next section, we develop an algorithm that exploits this relationship in order to find the true pose parameters of the true depth sensor.

IV. Proposed Algorithm

In this section, we explain the new calibration algorithm. A schematic visualization of this algorithm is given in Fig. 4. Essentially, three steps are performed. First, a uni-modal estimator is used to find an initial sensor pose according to (9). Second, from this first solution, 23 ambiguous pose parameters are calculated. For each of these pose parameters an additional uni-modal estimator is launched with a random walk system model (6), assuming a constant pose for each of the ambiguous sensors. Aggregated, these individual estimators can be seen as a multi-modal estimator. An additional discrete estimator then rates which of the 24 parallel estimators fits best to the constant pose model. Once, one of the estimator is considered to be converged to the true sensor pose, this estimator performs a refinement while all other estimators are stopped. In the following, the three steps are explained in more detail.

A. Initialization

In the first step, an uni-modal estimator (e.g., UKF) is set up with arbitrary prior state $\underline{\boldsymbol{x}}_0^s \sim \mathcal{N}(\underline{\hat{x}}_0^s, \mathbf{C}_{x_0}^s)$, e.g., $\underline{\hat{x}}_0^s = \underline{0}$ and high uncertainty $\mathbf{C}_{x_0}^s$. After a number of time steps n, this estimator converges to a single local solution $\underline{\boldsymbol{x}}_n^s \sim \mathcal{N}(\underline{\hat{x}}_n^s, \mathbf{C}_{x_n}^s)$ that is considered to be one of the 24 ambiguous solutions $\underline{\hat{x}}_n^{s,1}, \ldots, \underline{\hat{x}}_n^{s,24}$.

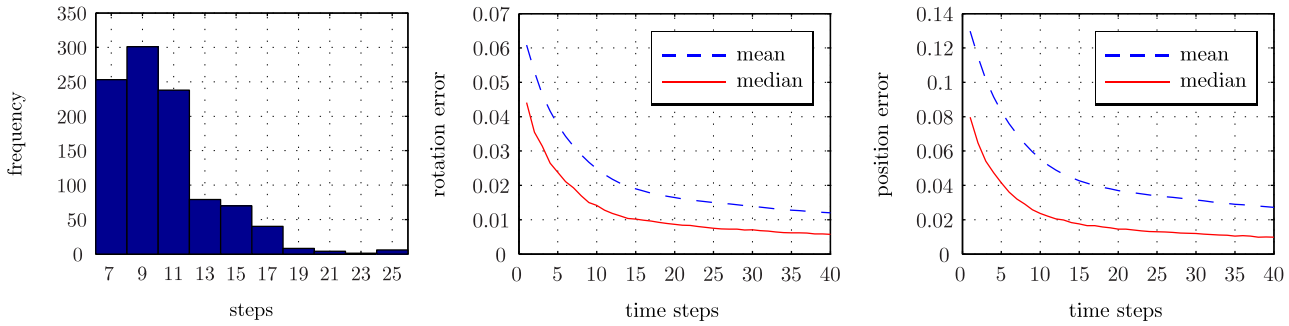
B. Resolve Ambiguities

The ambiguous solutions $\underline{\hat{x}}_n^{s,1}, \ldots, \underline{\hat{x}}_n^{s,24}$ are calculated from $\underline{\hat{x}}_n^s$ by applying the formulas from Appendix A. Then, 24 uni-modal estimators (again UKF) are launched in parallel, where each $i = 1, \ldots, 24$ is initialized with the prior state distribution $\underline{\boldsymbol{x}}_n^{s,i} \sim \mathcal{N}(\underline{\hat{x}}_n^{s,i}, \mathbf{C}_{x_n}^{s,i})$. As the cube moves, the undesired parameters constantly change while the desired parameters converge to static values. In order to exploit this relation, a multiple model approach is used. A probability $p_k^{s,i} = \frac{1}{24}$ for each estimators is introduced that represents the confidence that i the desired result. After each measurement update this model probability is also updated according to Bayes' rule

$$p_{k+1}^{s,i} \propto b_{k+1}^{s,i} \cdot p_k^{s,i} \quad . \tag{10}$$

where $b^{s,i}$ acts as the likelihood

$$b^{s,i} = \mathcal{N}(0; E\{\mathrm{g}(\mathbf{H}^{s,i} \cdot (\underline{\hat{y}}^s - \underline{\boldsymbol{w}}^s), \underline{c})\}, Var\{\mathrm{g}(\mathbf{H}^{s,i} \cdot (\underline{\hat{y}}^s - \underline{\boldsymbol{w}}^s), \underline{c})\}) \quad , \tag{11}$$



(a) Decision steps to resolve ambiguities. (b) Rotation error after refinement. (c) Position error after refinement.

Fig. 5: Result of 1000 Monte-Carlo runs of the poroposed algorithm. (a) shows a histogram of required decision steps for resolving the symmetric ambiguities. In (b,c), the average (median) Euclidian distance to the ground truth is depicted.

that rates how well the measurements fit to the estimated pose parameters. It is important to note that mean E and variance Var in (11) are intermediate results in the employed UKFs, such that no additional calculations are required. Note that this multiple model approach can be seen as a special case of an *interacting multiple model* (IMM) [20], where the probability of model transitions are zero. The parallel estimation of all 24 models is stopped at a time step N, when one model i exceeds a defined probability threshold Θ, so that $\max(p_N^{s,i}) > \Theta$. The corresponding estimate $\underline{\boldsymbol{x}}_N^{s,i}$ is then assumed to be the desired one.

C. Refinement

After dropping all other estimators, the desired estimator i is continued without applying any time update until convergence.

V. Evaluation

In order to evaluate the proposed algorithm, we conducted two experiments with synthetic and real data.

A. Synthetic Data

For synthetic evaluation, 1000 Monte-Carlo runs of the calibration algorithm were performed, each of them with a different, randomly generated depth sensor pose. A simulated cube with edge length 30 cm was moved along the edge of a square in the xy-plane. This straight, orthogonal motion was considered to be feasible for most positioning units. Each time step 15 measurements from the cube were generated by means of the Kinect sensor model, with respect to visibility constraints. Please note that we intentionally chose the number of measurements far lower as it would have been in a real calibration scenario, in order to emphasize that the proposed algorithm can also be applied to low resolution sensors.

Then, the simulated measurements and the corresponding global cube pose were provided as input to the calibration algorithm. For initialization we chose $\underline{\boldsymbol{x}}_0^s \sim \mathcal{N}(\underline{0}, \mathrm{diag}(10^{-3}, 10^{-3}, 10^{-3}, 1, 1, 1))$ and performed $n = 40$ estimation steps with an UKF. Subsequently, 24 UKFs were started for the ambigious poses. For the convergence criterion, we chose $\max(p^{s,i}) > 0.999$. For refinement, 40 steps were performed. The results of the experiments are shown in Fig. 5. On average, it took 10.72 steps to resolve all symmetric ambiguities (see Fig. 5a). The statistics of the calibration quality are shown in in Fig. 5b-c. The median position error after refinement is less then 0.99 cm. The mean is about 2.73 cm, due to the fact that randomly generated depth sensor poses may have been far away from the cube object. Note that no further optimizations based on, e.g., ICP algorithms were performed.

B. Real Data

For the real data experiments, we employed a depth sensor network that consists of three Kinect depth sensors. A *blue* cube was attached to a positioning unit that moved the cube and provided accurate pose data. The experimental environment is depicted in Fig. 1. We used a depth sensor configuration with overlapping fields of view to allow for comparison with a standard checkerboard calibration method [21] that is considered as ground truth. The cube was extracted from the measured point clouds of each sensor by means of color segmentation in the color image of the Kinect. The resulting calibration is shown in Fig. 6. The average position error of 3.19 cm and 0.047 rotation error agree with the synthetic results.

VI. Conclusions

In this paper, we presented a new approach to extrinsic calibration of a depth sensor network. The proposed algorithm is based on measuring the surface of a mobile 3D object with known pose. A simple cube object was chosen, in order to deal with the strongly varying measurement quality. This quality was incorporated by means of a sensor model. In addition, the measurement model allows for processing partial measurements, caused by self-occlusion effects. The evaluation showed a very good performance, with simulated data, as well as with real data. The position accuracy is about the order of 1 cm without any post processing optimization.

A. Future Work

Future work will include implementation of the proposed algorithm in the presented telepresence setup and an extensive

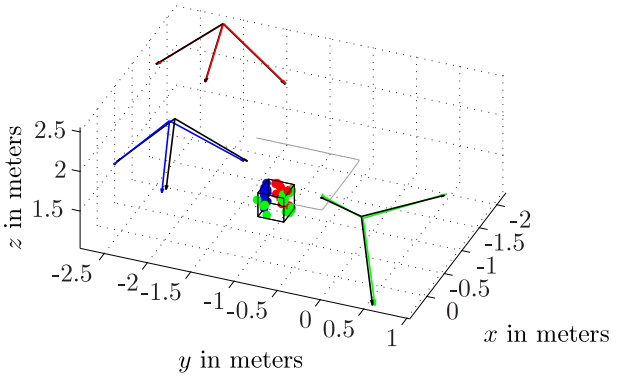


Fig. 6: Result of the real data calibration. The *black* poses are obtained by the checkerboard method. The result of the proposed algorithm and example measurements are drawn in *red*, *green*, and *blue*. The trace of the cube is drawn in *gray*.

evaluation in everyday life. Additionally estimating the pose of the calibration object would allow for far more flexible setups, but would come with the increasing complexity of a SLAM problem. The final goal would be simultaneously tracking a person and calibrating the sensor network.

Appendix

A. Ambiguous Depth Sensors

Let a cube be given with pose $\mathbf{H}^c$ and a depth sensor with pose $\mathbf{H}^s$. The 24 ambiguous depth sensor poses can be derived according to

$$
\begin{aligned}
\mathbf{H}^{s,1} &= \mathbf{H}^s, & \mathbf{H}^{s,2} &= \mathbf{H}^c\mathbf{H}_\beta(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,3} &= \mathbf{H}^c\mathbf{H}_{\beta\beta}(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,4} &= \mathbf{H}^c\mathbf{H}_{\beta\beta\beta}(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,5} &= \mathbf{H}^c\mathbf{H}_\gamma(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,6} &= \mathbf{H}^c\mathbf{H}_\alpha\mathbf{H}_\gamma(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,7} &= \mathbf{H}^c\mathbf{H}_{\alpha\alpha}\mathbf{H}_\gamma(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,8} &= \mathbf{H}^c\mathbf{H}_{\alpha\alpha\alpha}\mathbf{H}_\gamma(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,9} &= \mathbf{H}^c\mathbf{H}_{\gamma\gamma}(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,10} &= \mathbf{H}^c\mathbf{H}_\beta\mathbf{H}_{\gamma\gamma}(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,11} &= \mathbf{H}^c\mathbf{H}_{\beta\beta}\mathbf{H}_{\gamma\gamma}(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,12} &= \mathbf{H}^c\mathbf{H}_{\beta\beta\beta}\mathbf{H}_{\gamma\gamma}(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,13} &= \mathbf{H}^c\mathbf{H}_{\gamma\gamma\gamma}(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,14} &= \mathbf{H}^c\mathbf{H}_\alpha\mathbf{H}_{\gamma\gamma\gamma}(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,15} &= \mathbf{H}^c\mathbf{H}_{\alpha\alpha}\mathbf{H}_{\gamma\gamma\gamma}(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,16} &= \mathbf{H}^c\mathbf{H}_{\alpha\alpha\alpha}\mathbf{H}_{\gamma\gamma\gamma}(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,17} &= \mathbf{H}^c\mathbf{H}_\alpha(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,18} &= \mathbf{H}^c\mathbf{H}_\gamma\mathbf{H}_\alpha(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,19} &= \mathbf{H}^c\mathbf{H}_{\gamma\gamma}\mathbf{H}_\alpha(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,20} &= \mathbf{H}^c\mathbf{H}_{\gamma\gamma\gamma}\mathbf{H}_\alpha(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,21} &= \mathbf{H}^c\mathbf{H}_{\alpha\alpha\alpha}(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,22} &= \mathbf{H}^c\mathbf{H}_\gamma\mathbf{H}_{\alpha\alpha\alpha}(\mathbf{H}^c)^{-1}\mathbf{H}^s,\\
\mathbf{H}^{s,23} &= \mathbf{H}^c\mathbf{H}_{\gamma\gamma}\mathbf{H}_{\alpha\alpha\alpha}(\mathbf{H}^c)^{-1}\mathbf{H}^s, & \mathbf{H}^{s,24} &= \mathbf{H}^c\mathbf{H}_{\gamma\gamma\gamma}\mathbf{H}_{\alpha\alpha\alpha}(\mathbf{H}^c)^{-1}\mathbf{H}^s,
\end{aligned}
$$

where $\mathbf{H}_{xx} = \mathbf{H}_x\mathbf{H}_x$, $\mathbf{H}_{xxx} = \mathbf{H}_x\mathbf{H}_x\mathbf{H}_x$, and

$$
\mathbf{H}_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & \cos\phi & -\sin\phi & 0\\ 0 & \sin\phi & \cos\phi & 0\\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{H}_\beta = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0\\ 0 & 1 & 0 & 0\\ -\sin\phi & 0 & \cos\phi & 0\\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
\mathbf{H}_\gamma = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0\\ \sin\phi & \cos\phi & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{with } \phi = \frac{\pi}{2}.
$$

References

[1] F. Faion, S. Friedberger, A. Zea, and U. D. Hanebeck, "Intelligent Sensor-Scheduling for Multi-Kinect-Tracking," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura, Algarve, Portugal, Oct. 2012, pp. 3993–3999.

[2] K. Berger, K. Ruhl, C. Brümmer, Y. Schröder, A. Scholz, and M. Magnor, "Markerless Motion Capture Using Multiple Color-Depth Sensors," in *Proc. Vision, Modeling and Visualization (VMV)*, Berlin, Germany, Oct. 2011, pp. 317–324.

[3] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab." [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[4] S. Fuchs and G. Hirzinger, "Extrinsic and Depth Calibration of ToF-Cameras," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Anchorage, Alaska: IEEE, Jun. 2008, pp. 1–6.

[5] D. Herrera C, J. Kannala, J. Heikkila, and D. H. C., "Joint Depth and Color Camera Calibration with Distortion Correction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2058–2064, May 2012.

[6] T. Svoboda, D. Martinec, and T. Pajdla, "A Convenient Multi-Camera Self-Calibration for Virtual Environments," *Presence: Teleoperators & Virtual Environments*, vol. 14, no. 4, pp. 407–422, 2005.

[7] B. Kainz, S. Hauswiesner, and G. Reitmayr, "OmniKinect: Real-Time Dense Volumetric Data Acquisition and Applications," in *Virtual Reality Software and Technology (VRST)*, Toronto, Ontario, Canada, 2012.

[8] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion," in *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, no. Icvs, New York, NY, USA, 2006, pp. 45–45.

[9] K.-Y. K. Wong, G. Zhang, and Z. Chen, "A stratified approach for camera calibration using spheres." *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 20, no. 2, pp. 305–16, Feb. 2011.

[10] Z. Chen, D.-C. Tseng, and J.-Y. Lin, "A simple vision algorithm for 3-D position determination using a single calibration object," *Pattern Recognition*, vol. 22, no. 2, pp. 173–187, Jan. 1989.

[11] R. Hartley, E. Hayman, L. de Agapito, and I. Reid, "Camera Calibration and the Search for Infinity," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1. Toronto, Ontario, Canada: IEEE, 1999, pp. 510–517 vol.1.

[12] H. K. Stern, W. Yonescu, and D. A. Briceno, "Calibration of Three-Dimensional Space," 1990.

[13] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, a. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. Basel, Switzerland: Ieee, Oct. 2011, pp. 127–136.

[14] F. Faion, P. Ruoff, A. Zea, and U. D. Hanebeck, "Recursive Bayesian Calibration of Depth Sensors with Non-Overlapping Views," in *Proceedings of the 15th International Conference on Information Fusion (Fusion 2012)*, Singapore, 2012.

[15] F. Faion, M. Baum, and U. D. Hanebeck, "Tracking 3D Shapes in Noisy Point Clouds with Random Hypersurface Models," in *Proceedings of the 15th International Conference on Information Fusion (FUSION)*, Singapore, 2012, pp. 2230–2235.

[16] A. P. Arias and U. D. Hanebeck, "A Novel Haptic Interface for Extended Range Telepresence: Control and Evaluation," in *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2009)*, Milan, Italy, 2009, pp. 222–227.

[17] I. Fukui, "TV image processing to determine the position of a robot vehicle," *Pattern Recognition*, vol. 14, no. 1-6, pp. 101–109, Jan. 1981.

[18] F. Faion, A. Zea, M. Baum, and U. D. Hanebeck, "Partial Likelihood for Unbiased Extended Object Tracking," in *Proceedings of the 18th International Conference on Information Fusion (Fusion 2015)*, Washington D. C., USA, 2015.

[19] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators," *Automatic Control, IEEE Transactions on*, vol. 45, no. 3, pp. 477–482, 2000.

[20] H. A. P. Blom and Y. Bar-Shalom, "The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients," *Automatic Control, IEEE Transactions on*, vol. 33, no. 8, pp. 780–783, Aug. 1988.

[21] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.