Calibrating Low-cost Structured-light 3D Sensors

R. Chakib
Université de Limoges
XLIM / ASALI
123 Av. Albert Thomas
87000 Limoges, France

CORUO
reda.chakib@etu.unilim.fr

N. Mérillou
Université de Limoges
XLIM / ASALI
123 Av. Albert Thomas
87000 Limoges, France
nicolas.merillou@unilim.fr

P.-J. Vincent
CORUO
46 Av. des Bénédictins
87000 Limoges, France
pierre-jean.vincent@coruo.com

S. Mérillou
Université de Limoges
XLIM / ASALI
123 Av. Albert Thomas
87000 Limoges, France
stephane.merillou@unilim.fr

ABSTRACT

Consumer-grade RGB-D cameras are widely accessible, but they suffer from a lack of accuracy when compared to professional-grade 3D scanning solutions. In this paper, we propose a new method for calibrating an Intel RealSense SR300 camera, adaptable to other structured light sensors. The method uses classical checkerboard calibration and a coordinate-measuring machine (CMM) based setup with a calibrating plane. It delivers better results than the manufacturers settings.

Keywords

Camera calibration, RGB-D camera, coordinate-measuring machine, pinhole model, intrinsic calibration.

1 INTRODUCTION

Despite being widely accessible and user-friendly, low-cost structured light cameras suffer from a major problem related to their accuracy. The manufacturers generally use proprietary calibration methods with their devices, which leads to semi-closed technologies. Therefore, experienced end-users cannot benefit from the full potential of their sensors. A proper calibration may lead to a better precision when compared with the factory default settings.

The introduction of the Microsoft Kinect was the beginning of the era of consumer grade RGB-D cameras. Then the Intel RealSense sensors line introduced efficient, compact and easily embeddable devices. We chose to work with the Intel RealSense SR300, which covers short-range areas. This camera contains a color sensor, an IR sensor and an IR projector for depth measurement. The onboard imaging chip processes the depth computation [Int16].

In use, the RealSense SR300 presents some inaccuracies, for example when capturing a flat wall, the point cloud is warped at the corners, see Fig. 1. The IR sensor also suffers from distortion at the edges of the IR frames as shown in Fig. 2.

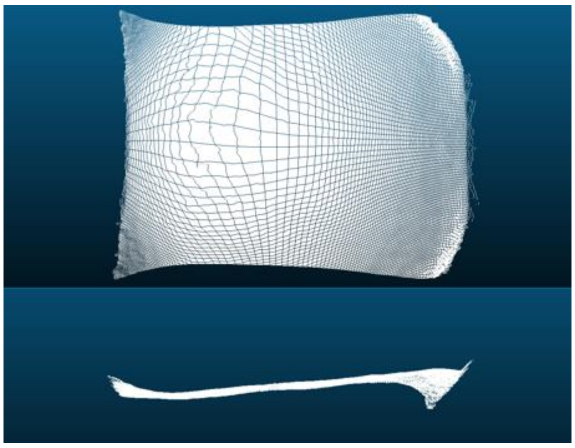


Figure 1. Point cloud of a flat surface captured using the SR300 with default settings.

This paper describes a new calibration method for the Intel RealSense SR300 with a twofold achievement:

- Improving the accuracy over the manufacturer's calibration;
- Providing a general-purpose calibration method that can be applied to similar devices;

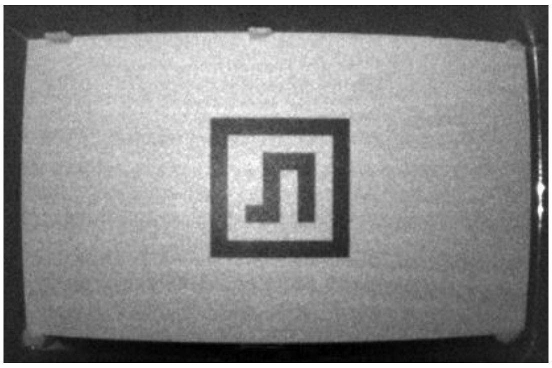


Figure 2. Distortion in the SR300 IR. The panel with the pattern is rectangular.

Our algorithm consists in two main steps:

- A classic checkerboard calibration or 2D calibration to correct the camera rays (IR camera).
- A depth correction performed using a Coordinate-Measuring Machine (CMM) for high precision measurement.

The output is a calibration data file with the camera parameters and a 3D grid of correction coefficients covering the calibration domain in the view frustum of the depth camera.

This paper is organized as follows. Section 2 gives a brief introduction of the camera's intrinsic parameters, then it presents related works about RGB-D cameras calibration. Section 3 presents our method and provides all the details on the hardware setup. Section 4 contains some experimental results along with a validation approach for our method. Finally, Section 5 is a discussion/conclusion on our work.

2 BACKGROUND AND RELATED WORK

Camera calibration is the process of mathematically describing how 3D spatial points project into the camera image sensor. That is, a mathematical model of the camera is required for calibration. We use the pinhole camera model for the camera's parameters description.

2.1 Camera's Intrinsic Parameters

The pinhole camera model describes the projection of 3D world points into the camera's (2D) image plane.

Let us consider a point $M_c = [x_c, y_c, z_c]^T$ in the camera frame. We want to express the projection of M_c using image coordinates which we denote $P_c = [u_c, v_c]^T$ using the pinhole model.

First, we begin by normalizing the point M_n:

$$M_n = [x_n, y_n]^T = [x_c/z_c\,, y_c/z_c]^T.$$

In the pinhole camera model, the rays are considered to pass linearly through the optical center, which in the case of real cameras is not true. In fact, the use of lenses alters the linearity of the light rays which causes non-linear distortion on the final images.

Using the normalized point, the distortion is performed in two steps [HKH12]:

$$M_g = \begin{bmatrix} 2k_3x_ny_n + k_4(r^2 + 2x_n^2) \\ k_3(r^2 + 2y_n^2) + 2k_4x_ny_n \end{bmatrix}$$

$$M_k = (1 + k_1r^2 + k_2r^4 + k_5r^6)M_n + M_g$$

where $r^2 = x_n^2 + y_n^2$ and $k_c = [k_1, \dots, k_5]$ is the vector of the distortion coefficients.

The point Pc that we are looking for is:

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} f_{cx} & 0 \\ 0 & f_{cy} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} u_{0c} \\ u_{0c} \end{bmatrix}$$

The parameters $\{f_{cx}, f_{cy}, p_{0c}, k_1, k_2, k_3, k_4, k_5\}$ are called the *intrinsic parameters of the camera* where $\{f_{cx}, f_{cy}\}$ are the *focal lengths* and $p_{0c} = [u_{0c}, v_{0c}]$ is the camera *principal point*. Intrinsic calibration consists in finding these parameters. To do so, we should establish the correspondence between a set of 3D points and their projected 2D image points [Sem16].

Zhang [Zha04a] made the following classification for calibration techniques, based on the dimensionality of the calibration object:

1) 3D reference object-based calibration: the typical 3D calibration object is composed of two or three orthogonal planes [Hei00]. The geometry of the object should be known with high precision.

2) 2D plane-based calibration: consists in using a planar object such as a checkerboard or circular patterns printed on a panel captured from different point of views. Many resources are available on the subject [Zha00], [SM99].

3) 1D line-based calibration: first proposed by Zhang [Zha04b], it consists in observing a set of collinear points moving around a fixed point.

4) Self-calibration: or 0D calibration as referred to by Zhang [Zha04a] because no calibration object is required. The method consists in calibrating the camera form a sequence of images of a static scene, without any prior knowledge of the camera's motion [HZ05].

2.2 Depth Cameras Calibration

Although built around the Kinect v1 sensor, most of the methods that we cite are supposed to be compatible with a wide range of low cost structured light cameras according to their respective authors. When the calibration object is known (shape, color, size), the calibration method is said to be *supervised*. Otherwise, the method is called *unsupervised*.

Smisek *et al*. [SJP11] proposed a geometrical model for the Kinect v1 and estimated the intrinsic parameters of both the IR and RGB cameras as well as their relative pose. They also estimated internal parameters of the depth camera. They used a checkerboard as the calibration target of both the RGB and IR cameras of the Kinect.

Herrera *et al*. [HKH12] have used a high-resolution color camera rigidly attached to the Kinect to compensate for the Kinect lower resolution color sensor. The calibration target is a planar board where a checkerboard is printed or stuck. In addition to the intrinsic parameters and the relative pose, the authors estimated the depth camera intrinsics.

Jin et al. [JLG14] have performed an intrinsic calibration of a Kinect unit, using a set of well-manufactured cuboids as their calibration target. Their objective function is a linear combination of the distance and angle errors from the cuboid. They re-wrote the objective function in terms of the intrinsic parameters of the camera prior to the optimization step.

Staranowicz et al. [SBMM15] have used a video of a spherical object moving in front the camera as input to their method. After a robust feature-extraction process, their algorithm infers an initial estimation of the depth, as well as the other calibration parameters, and then it performs a refinement estimate of the different parameters.

3 CALIBRATION METHOD

Our technique works as follow. First, the camera's intrinsic parameters are computed via a classical checkerboard approach, to correct the *x* and *y* coordinates. Then, the sensor is mounted on a CMM in front of the measure plane. Successive captures of the plane are acquired while moving towards it by using the corrected model from the first step. Then, we compute a 3D grid of correction coefficients that we infer from the collected data (plane's captures).

We could have dropped the checkerboard step, and instead rotated the plane by 45 degrees at each of its axis, but the errors in each direction would mix up. An alternative would be also to drop the checkerboard calibration, and to capture a calibrating sphere at different positions, then compute the errors, but we would be using inaccurate captures as we rely on the manufacturer's calibration.

3.1 2D Calibration

As previously said, to get more accurate camera's intrinsic parameters (i.e. in order to remove the distortion shown in Fig. 2) we use a classical checkerboard calibration. We photograph a checkerboard from different viewpoints using the camera, and simply use OpenCV calibration module to compute the camera's parameters, in our case we are interested in the intrinsic values of the IR sensor.

Practically, we use the intrinsic values to compute the point's coordinates. The relationship between a 3D point (x, y, z) in space and its correspondent (u, v) in the depth image is as follows:

$$x = \frac{(u - p_x)z}{f_x}$$

$$y = \frac{(v - p_y)z}{f_y}$$

Where: (f_x, f_y) is the focal distance and (p_x, p_y) the optical center coordinates. The coordinate z is the depth that the sensor returns for the depth image pixel (u, v).

Finally, we apply on x and y a similar iterative distortion compensation scheme to the one used in OpenCV. The correction over the X and Y axes is equivalent to correcting the camera's ray directions.

Now, we need to adjust the position of each acquired point all along its corresponding camera ray.

3.2 Depth Calibration

At this step, we compute a regular 3D grid of correction coefficients over the view frustum of the sensor (a truncated pyramid) or a part of it. A set of captures of a calibrating plane is used to "feed" the grid's nodes in terms of point correction.

The process consists in two main steps:

- *Data acquisition*: "Real" points spread over the calibration domain and their correction.
- *Grid definition and nodes filling*: "Virtual" points embedding the local correction information and regularly spread over the calibration domain.

For a given sensor, these steps are performed only once to define its proper correction grid.

3.2.1 Data Acquisition

The input data is a set of points, captured by the sensor that we want to calibrate spread over the calibration domain which is the subspace defined by the correction grid. Every point should have a correction coefficient.

To this end, we used a matte white plane with a marker printed on its center. We place our plane against the inner panel of the CMM as shown in Fig. 3. We adjust the sensor's orientation so that it sits parallel to the calibrating plane (more details about the plane and the sensor adjustments are given in section 3.4). Successive captures of the plane are acquired by starting from the farthest distance in the calibrating domain and moving the sensor towards the plane with a fix step until the whole domain is covered.

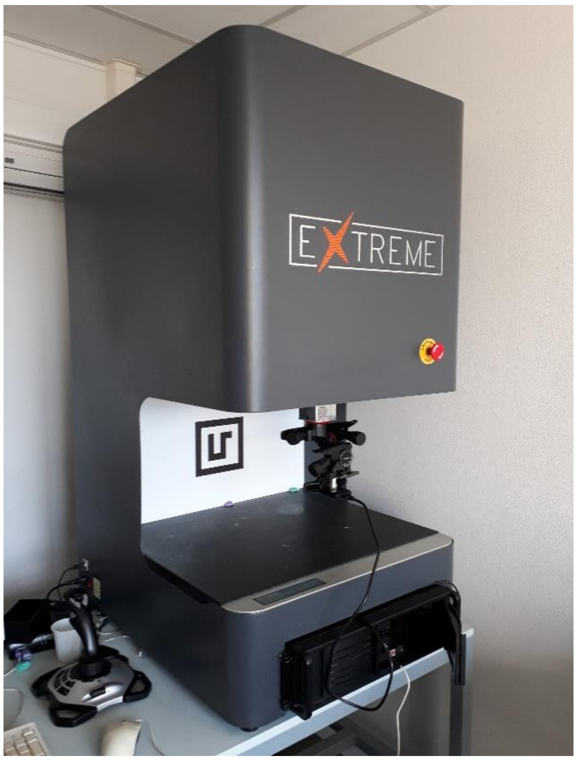


Figure 3. The calibrating plane and its setup on the CMM.

The 2D calibration process corrected the X and Y coordinates, that is the camera rays. Therefore, for every acquired point (of the plane), the correction coefficient we are looking for should slide the point back or forth along the camera ray so that the point's depth matches the real depth. In other words, we are looking for the real distance between the plane and the sensor to compute the correction coefficient.

To compute the real distance between the plane and the sensor, we use image processing to detect the marker printed on the calibrating plane and we apply the similar triangles principle using the focal distance that we already computed with the checkerboard method. Once the first distance computed, we use the CCM in order to infer the next distances for the successive calibrating plane captures.

The correction coefficient of a given point *P* is equal to the real distance of the plane *tD(P)*, which is the true depth, divided by the depth returned from the sensor *sD(P)* as shown in Fig. 4. Therefore, the correction coefficient *c(P)* is:

$$c(P) = tD(P)/sD(P)$$

3.2.2 Grid Definition and Node Filling

3.2.2.1 Grid Definition

The 3D grid is a regular truncated pyramid shaped set of nodes over the calibration domain. Every node is a 4D vector such that the first three components are the nodes coordinates and the fourth component is the correction scalar corresponding to the node. The nodes are not actually sensor's acquired points, but rather "virtual" points embedding the correction information of their neighborhood.

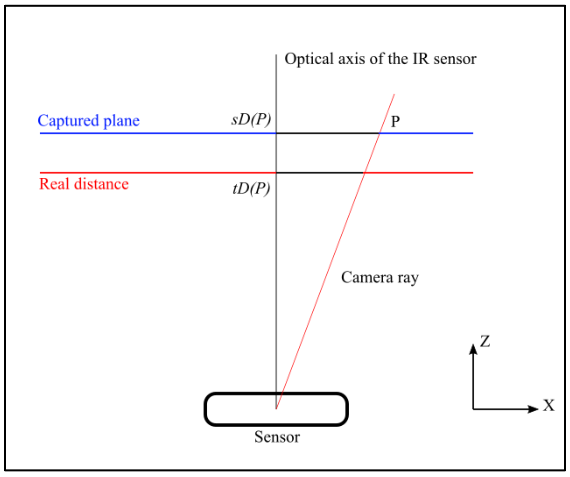


Figure 4. Correction coefficient for a given point *P*: the real depth of the calibrating plane *tD(P)* divided by the z coordinate of *P* returned by the sensor *sD(P)*.

The grid shape was chosen in order to guarantee a fair distribution of the points contributing to the correction computation in each node, regardless of the distance from the sensor.

We divide the Z-axis according to a fix step. We use the same step for capturing the calibrating plane with the couple sensor/CMM.

For the X-axis and Y-axis, we also use fixed steps. In addition, we take into account the maximum resolution of the depth sensor that we should not exceed.

Finally, it is important to consider the approximate number of points that will contribute to the correction of a node via interpolation.

3.2.2.2 Nodes Filling

The nodes positions are defined by the grid construction. Still, we need to compute the error correction in each node. To do so, we begin by defining the neighborhood of a node as all the cells that it belongs to. Using the points from the calibrating plane's captures, we interpolate every subset of points belonging to a neighborhood in order to compute its corresponding node's correction. In fact, each node embeds the correction information of the subspace defined by its neighborhood.

To interpolate over the defined neighborhoods, we used the inverse distance weighting interpolation method. It is defined as follows:

Let P the point to be corrected (the node), $\{P_i, i=1..N\}$ the vertices of its neighborhood, $d(P, P_i)$ the distance between the node P and the neighbor P_i, c_i the coefficient correction of the neighbor P_i, p a smoothing parameter and $c(P)$ the coefficient correction that we are looking for:

$$c(P) = \begin{cases} \sum_{i=1}^{N} \omega_i(P)\, c_i / \sum_{i=1}^{N} \omega_i(P), & if\ d(P, P_i) \neq 0\ \forall i \\ c_i, & if\ \ d(P, P_i) = 0\ for\ some\ i \end{cases}$$

Where:

$$\omega_i(P) = 1/d(P, P_i)^p$$

The smoothing parameter p controls the influence of far points on the interpolation. We took $p = 3$.

Once filled, the grid can be used to correct any point cloud captured within the subspace defined by it.

3.3 Applying the correction

In order to qualify for correction, a captured point cloud must belong partially or totally to the domain defined by the correction grid. That is, any point outside the calibration area cannot be rectified.

Let PC a point cloud captured with a calibrated sensor and G its correction grid. For every point P in PC, we start by finding the point's bounding cell BC in the grid G. Therefore, the inverse weighting interpolation can be applied across the nodes of BC to compute the correction for the point P. Finally, we multiply P by the computed coefficient to get a rectified point.

To determine the bounding cell of a given point, we define a 3D grid (a truncated pyramid) in which cells are numbered following *IJK* (K direction follows each ray from camera center over our domain). The coordinates *(i,j,k)* refer to the cell with the top-left-front vertex (from the point of view of the sensor. See Fig. 5.

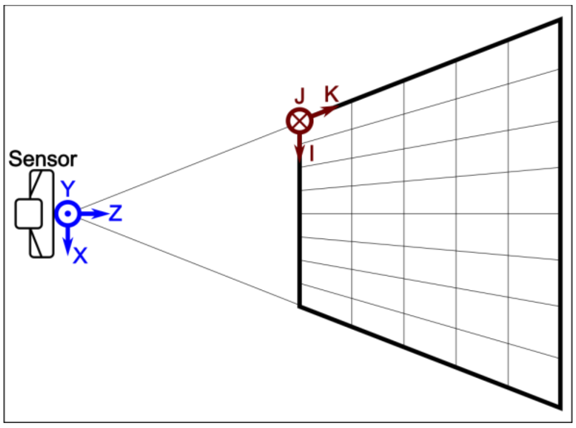


Figure 5. Top view of the newly defined 3D space, *IJK* (top view).

Therefore, beside the *(x,y,z)* coordinates of a given point *M(x,y,z)*, we just defined new coordinates *(i,j,k)* in the *IJK grid* which indicates the bounding cell of the point as follows:

1 - We start by finding *K*-coordinate. In fact, for a given *k*, all the nodes corresponding to the "level" *k* share the same depth. Thus, for every level, we can compare the current point's depth to the first node of each level starting from the farthest level to the sensor. The first level for which the first node's depth is less than the point's depth defines the *K* component. Thus, the bounding cell that we are looking for lays on that level.

2 - To find the *J*-coordinate, we restrict our search to the k^{th} level obtained from the previous step. We compute a signed angle between OM_{YZ} and Z-axis, where $OM_{YZ}(0,y,z)$ is the orthogonal projection of M on the plane *YZ*. We compare this angle against the signed angles computed between the projections on the plane *YZ* of the first node of each row from the level *k*, and the *Z*-axis.

3 - For the *I*-coordinate, we restrict our search to the k^{th} level obtained from the first step, and the j^{th} row obtained from the second step. We compute a signed angle between OM_{XZ} and *Z*-axis, where $OM_{XZ}(x,0,z)$ is the orthogonal projection of M on the plane *XZ*. We compare this angle against the signed angles computed between the projections on the plane *XZ* of each node of the j^{th} row from the k^{th} level, and the *Z*-axis.

3.4 Hardware Setup

We secure the calibrating plane against the inner panel of the CMM using modeling clay. In fact, it allows adjusting the plane, so it lays orthogonal to the *Y*-axis of the CMM. We attach a mechanical touch probe to the CMM and we "draw" a rectangle near the border of calibrating plane. The probe should touch the calibrating plane in the entire trajectory. If the test fails in some area of the plane, we compensate for the displacement of the calibrating plane using modeling clay. Fig. 6 shows our setup.

Once the calibrating plane is properly set, we detach the mechanical touching probe from the CMM and we attach the couple geared head/sensor instead. Then, we track the marker on the calibrating plane, and use the geared head to fine tune the sensor's orientation. To this end, we perform the detection on the IR camera stream and we highlight the marker's corners when they align over the X-axis or the Y-axis of the sensor in our software. We align the corners couple wise, for example top-left with top-right then top-left with bottom-left. That is, we perform the alignment one direction at a time (fig. 7).

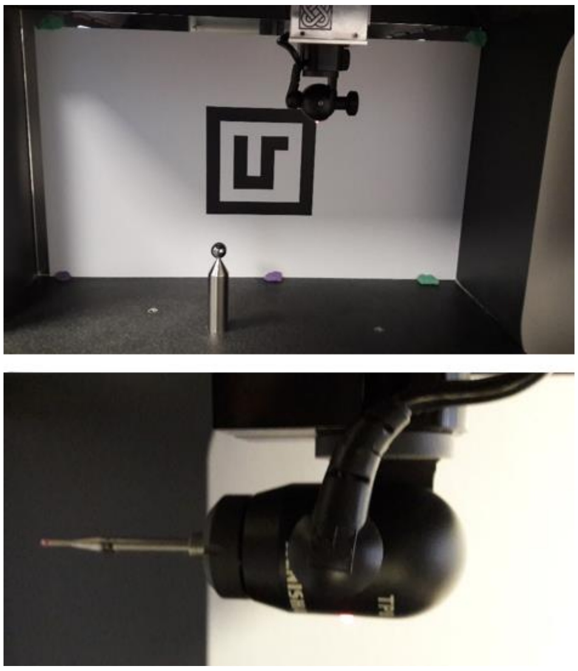


Figure 6. Top: the calibrating plane laying on the "inner panel" of the CMM. Bottom: the mechanical probe used to check the orthogonality of the plane with CMM Y-axis.

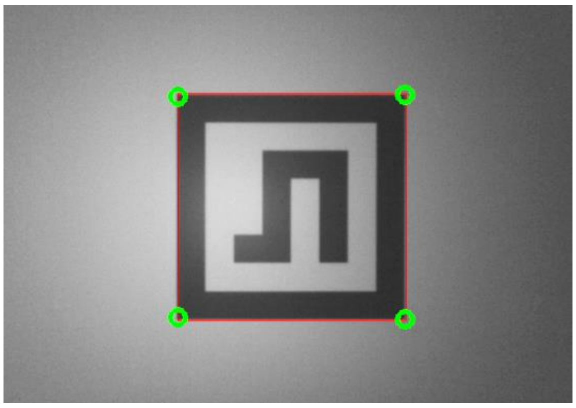


Figure 7. A real successful alignment; we used the green circles to highlight the aligned corners.

When the four corners of the marker align, meaning the sensor is parallel to the calibrating plane, we use the CMM joystick to move the sensor over the X-Y axes of the CMM so that the center of the marker matches the optical center of the sensor in the IR image. We recall that the optical center was computed during the checkerboard calibration. Therefore, we can apply the similar triangles principle to compute the ground truth distance.

In order to enhance the marker's detection, we turn off sensor's IR projector and use an external IR light source to illuminate the plane for a continuous IR illumination as the projector projects changing patterns.

Once the distance is measured, we spray a white matte powder to hide the marker in order to avoid the black color of the marker to distort theses points in the captured point cloud.

4 RESULTS AND VALIDATION

4.1 Calibration domain

According to the inner dimensions of the working space of the CMM, and for the calibrating plane to be fully covering the "frame" for each point cloud captured, we defined our calibration domain as the subspace of the depth view frustum located between 10 cm and 27 cm approximately from the IR camera center. The correction grid is of 64x48x50 size.

4.2 Checkerboard Calibration

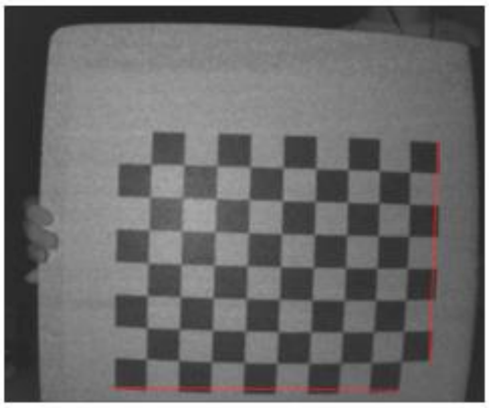
We performed a checkerboard calibration on the IR sensor giving the results on table 1. We took 48 pictures of a checkerboard using a 640x480 resolution. The checkerboard has 10x8 square tiles of 3 cm edges.

Fig. 8 shows a picture of the checkerboard before and after the correction via the computed distortion values. See Table I for the numerical results.

Parameter	Our values	Intel SDK's extracted values
Focal distances (pixels)	(473.448, 473.073)	(474.263, 474.263)
Principal point (pixels)	(308.148, 242.341)	(304.816, 245.449)
Radial distortion	(-0.117456, -0.0642003, 0.0390934)	(-0.120845, -0.0660312, 0.0516015)
Tangential Distortion	(-0.00148510, 0.000892128)	(-0.00265185, -0.00182552)
Average re-projection error	0.64	4.79

Table 1. The checkerboard calibration values vs sdk's

To compare the intrinsic parameters that we obtain against those of Intel's SDK, we use the re-projection error. Meaning, we re-project back feature points using the SDK's camera matrix and compare against the checkerboard reference positions, then we repeat the process using our camera matrix. In the end, we compute the average errors. See Table I for all the numerical values. Our computed parameters give a lower re-projection error than Intel's parameters.



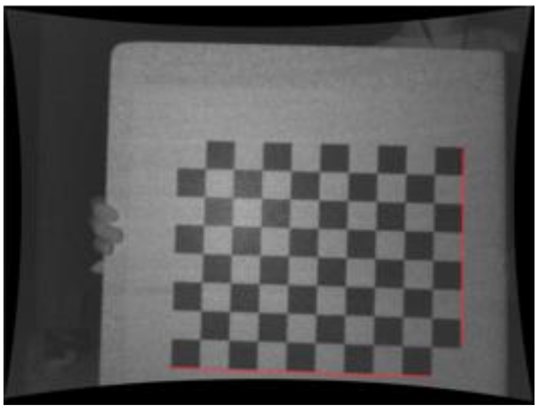


Fig. 8. On the top, a checkerboard picture without correction. On the bottom, the same picture after correcting the distortion. Straight red lines shows the distortion effect.

4.3 Depth Calibration

Before introducing our validation approach, we refer the reader to the in-depth RealSense SR300 assessment from a metrological point of view by Carfagni *et al.* [CFG+17]. Authors give an overview of the RealSense SR300 sensor capabilities and limits as a 3D scanning device.

Fig. 9. The calibration sphere used in our validation process: diameter 50.80 mm (2 inches).

Keeping the same hardware setup that we used for depth calibration, we replace the plane by a calibration sphere with a precisely known diameter Fig. 9. The goal is to capture the sphere at different positions of the calibration domain, then, estimate all the sphere centers using a best-fit approach to form a trajectory with the centers as nodes. For each capture or trajectory node, we acquire two point-clouds, one using the SDK's calibration values and the other using our calibrating values (checkerboard inferred intrinsic parameters). To the set of clouds captured using our values, we additionally apply depth correction.

We compute two errors per trajectory, a global error and a local error.

4.3.1 Global Error

For this estimator, no reference sphere is chosen, hence the term global. We denote the global error *E*.

We compute the distance of each sphere center to the next sphere center, in the order of their captures as no specific order is required. We will refer to the first set of distances as point cloud distances and we will denote it D_{PC}. Equivalently, we compute the distances between the successive CMM positions of the captures that we will call CMM distances and we will denote D_{CMM}. We define the global error as the following:

$$E = \sum_{\substack{d_{PC} \in D_{pc} \\ d_{CMM} \in D_{CMM}}} |d_{PC} - d_{CMM}| \,/(num\_spheres - 1)$$

Where: d_{CMM} is the correspondent of d_{PC} in D_{CMM}.

4.3.2 Local Error

A local error can be computed at each sphere center that we captured. For a sphere *S*, we compute the local error *e(S)* by taking the distances to all the other sphere centers and comparing them against the respective CMM inferred distances in a similar way of the global error. The local error at the sphere's center is:

$$e(S) = \sum_{\substack{d_{PC} \in D_{pc}(S) \\ d_{CMM} \in D_{CMM}(S)}} |d_{PC} - d_{CMM}| \,/(n\_spheres - 1)$$

Where $D_{PC}(S)$ is the set of distances computed from the point clouds and $D_{CMM}(S)$ is the set of distances computed from the respective CMM positions. d_{CMM} is the correspondent of d_{PC} in $D_{CMM}(S)$.

4.3.3 Results

We captured the calibrating sphere on twenty-seven different positions as shown in Fig. 10.

We recorded sets of three calibrations using our method under the same conditions. The plots in fig. 11 depict the global and local errors that we obtained. Although there are some positions where the RealSense SDK calibration performed better than our calibration, our average global error is lower in all the experiments, see Table II for the average global error of each experiment. Concerning local error, we can see that our calibration is much better than the SDK's in all the experiments.

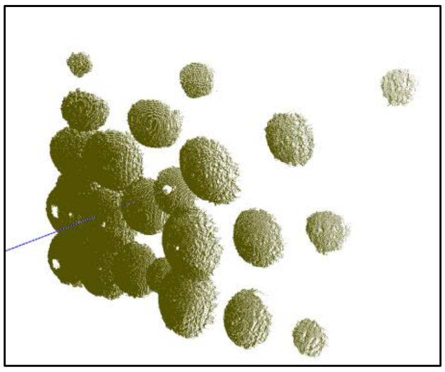
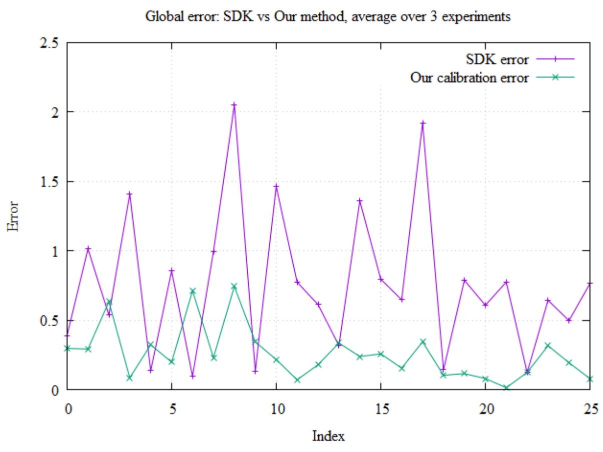


Fig. 10. The calibrating sphere captures over the calibration domain. The blue line corresponds to the Z-axis of the sensor.

Fig. 11. Shows a point cloud before and after the calibration. We chose a flat surface point cloud in order to see the actual difference. In fact, it is near the corners of a flat surface covering the whole "frame" that the distortion is mostly visible.



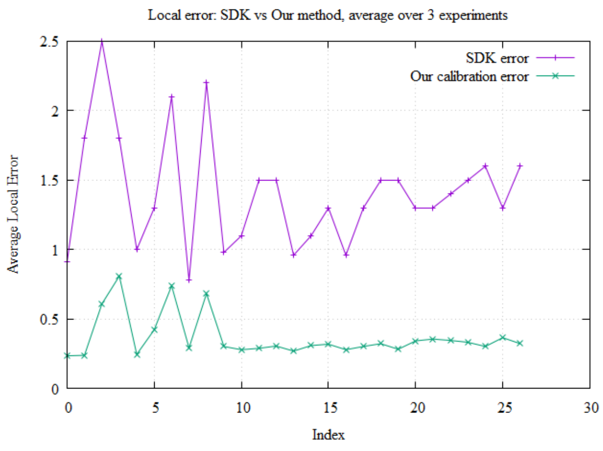


Fig. 11. Compared global error and local error plots SDK versus our method. We averaged over 3 experiments.

	Our calibration average error (mm)	SDK average error (mm)
1st experiment	0.18	0.76
2nd experiment	0.27	0.76
3rd experiment	0.32	0.76

Table 2. The global error evaluation

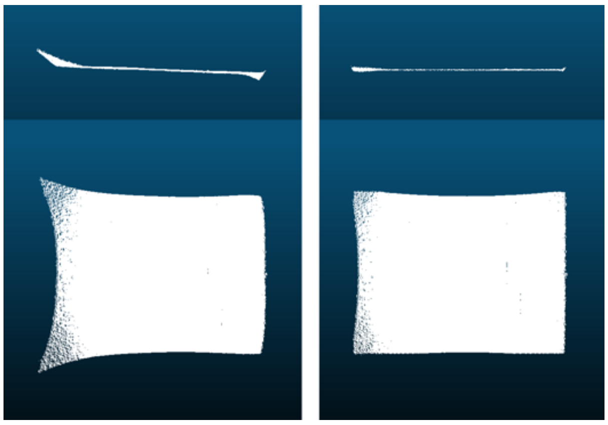


Fig. 12 Left: front and top view of a point cloud (flat surface) before correction. Right, the same plane after correction using our method.

4.3.4 Notes on the method's precision

The accuracy of our method essentially depends on two factors:

- The average re-projection error of the checkerboard calibration (see Table I). In our test, the error is 0.64 pixels.
- The precision of the ground-truth distance computed through image processing.

We will try to evaluate the second factor that is the accuracy of the ground-truth distance. It heavily relies on the average re-projection error as the corrected and undistorted IR frames are used in the image-processing step.

Using the similar triangles principle, the ground truth distance d is computed as follows:

$$d = \frac{L\,f}{l}$$

Where, L is the marker half-width (in millimeters), l is the marker half-width detected in the IR frame (in pixels) and f is the computed focal distance (in pixels) from the checkerboard calibration.

Now, suppose that we make a mistake of n pixels in our detection, and that the computed distance is d'. Then, the error corresponding to this detection is approximatively:

$$E(n) \approx \mathrm{d} - \mathrm{d}' = \frac{Lf}{l} - \frac{Lf}{l+n}$$

Thus,

$$E(n) \approx \frac{nLf}{l(l+n)}$$

The first thing to notice is that the bigger the value l, the smaller the error. To increase l, the IR camera should be set to its maximum resolution, that is 640x480 for the RealSense SR300, and the sensor should be very close to the camera in such a way that the marker cover most of the frame while still entirely enclosed in for detection sake.

To get an idea about the precision we achieved in our setup, we could get as close for a value of 225 pixels for l.

Knowing that $L = 79.5\,mm$ and $f = 473.448\,pixel$, the error is:

$$E(0.64\,pixels) \approx 0.47\,mm$$

Thus, we have approximatively a half millimeter precision in our ground truth distance.

5 CONCLUSION

In this paper, we have proposed a supervised intrinsic calibration method for the Intel RealSense SR300 that relies on the use of a CMM for robust ground truth. It has proven to give superior accuracy over the manufacturer's default calibration, as shown in the "Results and Validation" Section. In addition, we can apply it to other structured-light sensors, as we do not use any special or exclusive calibration parameter to the Intel RealSense SR300 sensor.

On the limitations side, when computing the X and Y coordinates, the method involves the use of a non-corrected yet depth coordinate (see equations page 3). Still, our approach performs better than the default manufacturer calibration, but as a future improvement, we will estimate the gap and if needed perform iterative calibration steps. On another side, we plan to make our method fully automated.

6 REFERENCES

[Int16]https://software.intel.com/sites/default/files/managed/0c/ec/realsense-sr300-product-datasheet-rev-1-0.pdf

[HZ05] Hartley, R., & Zisserman, A. (2005). Multiple view geometry in computer vision. Robotica, 23(2), 271-271.

[MVGV09] Moons, T., Van Gool, L., & Vergauwen, M. (2009). 3D reconstruction from multiple images, Part 1: Principles. Now Publishers Inc.

[HKH12] Herrera, D., Kannala, J., & Heikkilä, J. (2012). Joint depth and color camera calibration with distortion correction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(10), 2058-2064.

[Sem16] Semeniuta, O. (2016). Analysis of Camera Calibration with Respect to Measurement Accuracy. Procedia CIRP, 41, 765-770.

[Zha04a] Z. Zhang, "Camera Calibration", Chapter 2, pages 4-43, in G. Medioni and S.B. Kang, eds., Emerging Topics in Computer Vision, Prentice Hall Professional Technical Reference, 2004.

[Hei00] Heikkila, J. (2000). Geometric camera calibration using circular control points. IEEE Transactions on pattern analysis and machine intelligence, 22(10), 1066-1077.

[Zha00] Zhang, Z. (2000). A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence, 22(11), 1330-1334.

[SM99] Sturm, P. F., & Maybank, S. J. (1999). On plane-based camera calibration: A general algorithm, singularities, applications. In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. (Vol. 1). IEEE.

[Zha04b] Zhang, Z. (2004). Camera calibration with one-dimensional objects. IEEE transactions on pattern analysis and machine intelligence, 26(7), 892-899.

[SJP11] J. Smisek, M. Jancosek, T. Pajdla, 3D with Kinect, in: IEEE Workshop on Consumer Depth Cameras for Computer Vision, 2011.

[JLG14] Jin, B., Lei, H., & Geng, W. (2014, September). Accurate intrinsic calibration of depth camera with cuboids. In European Conference on Computer Vision (pp. 788-803). Springer International Publishing.

[SBMM15] Staranowicz, A. N., Brown, G. R., Morbidi, F., & Mariottini, G. L. (2015). Practical and accurate calibration of RGB-D cameras using spheres. Computer Vision and Image Understanding, 137, 102-114.

[CFG+17] Carfagni, M., Furferi, R., Governi, L., Servi, M., Uccheddu, F., & Volpe, Y. (2017). On the performance of the Intel SR300 depth camera: metrological and critical characterization. *IEEE Sensors Journal*, *17*(14), 4508-4519.