IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, Canada, September 2017.

Online Depth Calibration for RGB-D Cameras using Visual SLAM

Jan Quenzel, Radu Alexandru Rosu, Sebastian Houben and Sven Behnke

***Abstract*— Modern consumer RGB-D cameras are affordable and provide dense depth estimates at high frame rates. Hence, they are popular for building dense environment representations. Yet, the sensors often do not provide accurate depth estimates since the factory calibration exhibits a static deformation. We present a novel approach to online depth calibration that uses a visual SLAM system as reference for the measured depth. A sparse map is generated and the visual information is used to correct the static deformation of the measured depth while missing data is extrapolated using a small number of thin plate splines (TPS). The corrected depth can then be used to improve the accuracy of the sparse RGB-D map and the 3D environment reconstruction. As more data becomes available, the depth calibration is updated on the fly. Our method does not rely on a planar geometry like walls or a one-to-one-pixel correspondence between color and depth camera. Our approach is evaluated in real-world scenarios and against ground truth data. Comparison against two popular self-calibration methods is performed. Furthermore, we show clear visual improvement on aggregated point clouds with our method.**

I. Introduction

Modern consumer RGB-D cameras are affordable and provide dense depth estimates at high frame rates. Hence, these sensors are popular for building dense representations of indoor environment. Yet, the sensors often do not provide accurate depth estimates since the generation of depth estimates is mostly a black box which exhibits a sensor-dependent deformation. This bias can decrease the reconstruction quality and reduce the camera pose estimation accuracy, especially when using dense reconstruction frameworks like KinectFusion [1] or ElasticFusion [2], and may ultimately lead to failure of the tracking process.

The deformation can be present for a number of reasons. Wrong estimates on the calibration parameters for involved cameras can degrade the depth quality especially further away from the principal point. Different exposure and capture times of the cameras add an error depending on the current sensor motion. Another discrepancy exists between depth estimated from the moving RGB camera and IR-based depth estimates. Quality problems during manufacturing like a different housing, temperature changes, mechanical strain, or strong disturbances on the sensor parts also change the intrinsic parameters and depth estimates and may require a recalibration. On the upside, the strength of this deformation mostly depends on the position within the depth image and the range, which allows us to mitigate the effects by a better calibration. In contrast to existing approaches, we do not rely on fitting planes into planar surfaces of dense depth images [3], [4] in the recorded environment, or selecting only close measurements as reference [5] which makes our approach more versatile. For example, surface irregularities or a curved wall differ from the estimated plane and may bias the calibration. Furthermore, these dense approaches generate high computational load, even for a small number of used depth frames, and they disqualify for online calibration.

All authors are with the Autonomous Intelligent Systems Group, University of Bonn, Germany {quenzel,houben, behnke}@ais.uni-bonn.de

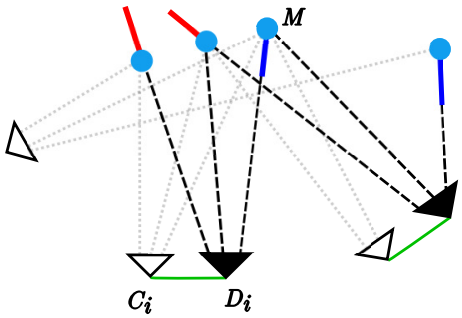
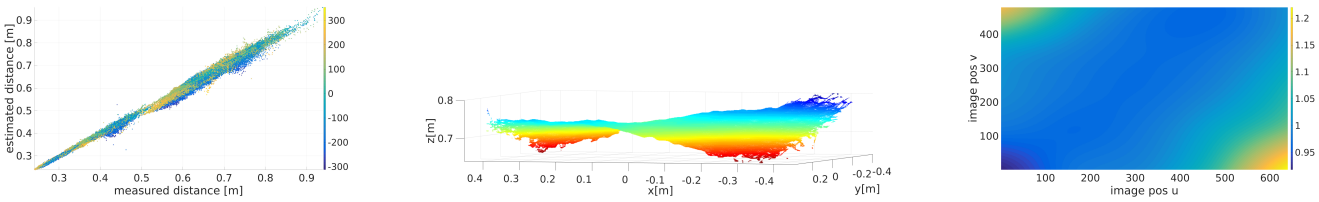


Fig. 1: Calibration principle: The map points M (blue dots) are estimated via triangulation from visual features (gray dotted lines) from RGB images C_i. Valid depth measurements (black dashed lines) are then compared against expected distance. Depending on the position within the depth image D_i, the depth is over- (red) or underestimated (blue).

We propose to use well-triangulated sparse visual map points as a reference to the measured depth from the depth sensor. Already a small number of such points allows us to calculate a very good estimate of the complete depth deformation. Using triangulated points has the further advantage that our method can be used in an arbitrary (textured) environment without the necessity for a calibration pattern. Additionally no manual parameter optimization has to be done for plane fitting. Instead of assuming a one-to-one relationship between color and depth pixels, we explicitly use the transformation between color and depth camera, allowing us to use an arbitrary sensor setup, like stereo and depth camera, and not restricting the method to RGB-D cameras. The only prerequisite for the applicability of our method is to have the intrinsic and extrinsic calibration for all cameras, obtained from an existing factory calibration or e.g. using a calibration toolbox like Kalibr [6]. Obviously, few points lead to a sparse coverage of the sensor, which makes interpolation necessary. For this, we assume that deformations of the depth map are locally similar and approximately smooth. This allows us to use methods of scattered data approximation like thin plate splines (TPS). For TPS parameter estimation we employ robust optimization.



(a) (b) (c)

Fig. 2: Depth distortion. (a) measured distance z_m vs. ground truth distance z_{gt} for the Intel RealSense SR300 camera used in our experiments. The colour encodes the distance of the corresponding pixel from the depth image center. (b) depth deformation on a planar floor, color-coded by distance to camera (increasing z direction). (c) estimated scaling in depth image coordinates.

II. Related Work

RGB-D camera calibration has been addressed by multiple research groups before. A typical approach consists in using an artificial calibration object like a checker board, which has good visual structure and planarity. This allows the end user to easily estimate the pose of the camera relative to the checker board or to fit a plane in the measured depth data. The Kalibr toolbox by Furgale et al. [6] can be used for the intrinsic and extrinsic calibration of multiple cameras with overlapping field of view using an artificial target. Unfortunately, this does not include depth cameras. A similar approach was taken by Herrera et al. [7], who calibrate two color cameras and a Kinect depth camera simultaneously with a checker board. They corrected depth using a spatially varying offset with exponentially decreasing weight for increasing range. Smisek et al. [8] calibrated the intrinsics of a Kinect camera and compared the results with a stereo rig. A depth-correction image was generated using the mean error from multiple planar measurements at different distances. The image was then subtracted from the measured depth image. Nguyen et al. [9] created a more accurate sensor model for a Kinect by explicitly modelling noise in terms of axial and lateral direction. They used a planar target which can be rotated w.r.t. the sensor.

There has also been work on improving the depth estimates of time-of-flight (ToF) cameras. Fuchs et al. [10], for example, use the monochrome and depth images in conjunction with a robotic arm for accurate positioning of the ToF sensor. The hand-eye-calibration between arm and camera is also included in their method, which optimizes all camera parameters in a non-linear fashion. Ferstl et al. [11] aim at calibrating color and depth simultaneously with a second color camera and a random regression forest for depth compensation. Again, a planar target with known size is used.

Depth calibration without artificial targets was done by Clarkson et al. [12], Teichman et al. [5] and di Cicco et al. [3]. Clarkson et al. [12] used a movable planar target to acquire calibration scans. A plane is fitted into these scans using PCA. The error between measurement and plane is estimated by a 2-degree polynomial for each pixel. Evaluation of the polynomial then yields the offset to correct the measurement.

Teichman et al. [5] showed that range-dependent *depth multiplier images* are an intuitive and effective method to correct the measured depth. The generation of these multipliers relies on a SLAM frontend, for trajectory and map estimation. A point cloud is created from only reliable close depth measurements given the trajectory. The desired map range for a specific pixel is averaged from nearby points within the point cloud. A maximum likelihood estimator (MLE) is then employed to generate each multiplier independently such that it minimizes the difference between map range scaled by a multiplier and the measured range. Calibration and SLAM could then be alternated for further improvement. Although the iteration was not found to be necessary for Kinect cameras. In contrast to our approach, the authors use binning of size 8×6 pixels in the image and create multiple depth multipliers for different ranges, resulting in 10^4 parameters. The actual factor for a depth measurement is then generated by linear interpolation between the depth multipliers at different ranges. Teichman et al. reported their algorithm to run for multiple minutes, which is too slow for online calculation, while the application of their correction takes only milliseconds for a depth image in VGA resolution. Basso et al. [4] use a calibrated color camera and an uncalibrated depth sensor. The extrinsic transformation between both is estimated using plane-to-plane constraints together with an undistortion map for the depth data based on a fitted plane. The whole optimization is reported to take approx. one hour for 400 frames at half Kinect resolution. Di Cicco et al. [3] use a similar undistortion model, as the scalar depth multiplier depends only on the pixel position and the measured range. In contrast to our approach or the approach by Teichman et al., no SLAM is used. Instead, the deformation is estimated from planar surfaces like walls under the assumption that the miscalibration produces smaller errors in the center of the depth image than close to the border. A plane is fitted into the central region. Inlier measurements from the whole frame are generated based on small difference between measured and plane distance along the viewing ray and a small normal angle. The approximated function is assumed to be smooth and continuous, but is coarsely discretized into a 3D matrix to get a more robust estimate while being easier to compute. The 3D matrix

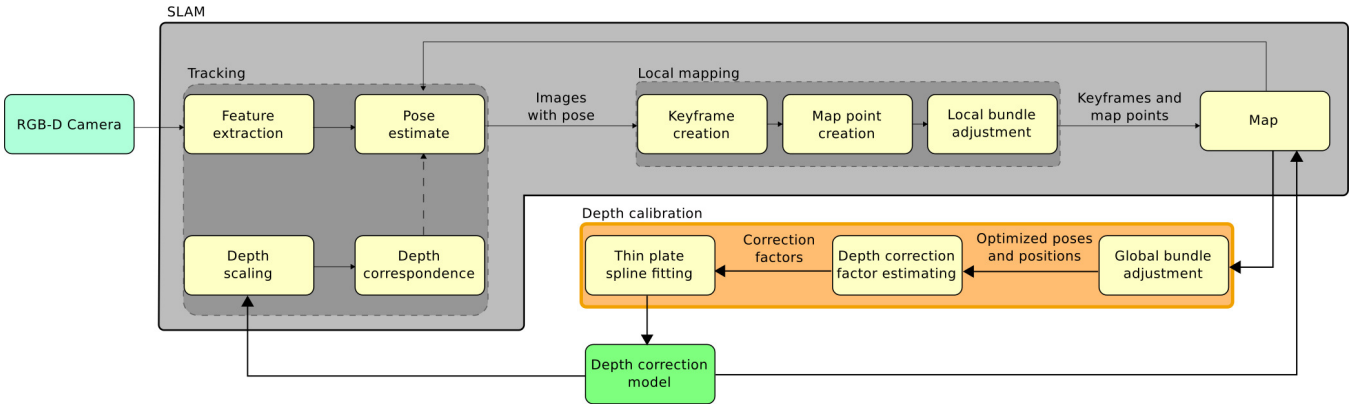


Fig. 3: Method overview: Visual features are tracked from RGB-D images in order to estimate the camera pose w.r.t. the map points. Keyframes and triangulated map points are computed by the local mapping and stored in the map. The depth calibration (orange dashed box) first performs a global bundle adjustment to optimize the keyframe poses and map point positions based on visual information only. Sparsely distributed pixel-wise depth correction factors are then estimated from projecting the map points into the depth images. A thin plate spline is fitted to obtain a dense depth correction model (green) that can be used to correct the map and to rescale measurements in depth images during tracking.

corresponds to multiple depth multiplier images at different ranges. To obtain such a 3D-matrix, the authors proposed a k-nearest-neighbour-based approach for offline scenarios. Missing data is filled in by averaging over the k closest data points. For online use cases, di Cicco et al. trained an artificial neural network using sparsely distributed depth measurements.

In the case of the PrimeSense sensors, the assumption of close depth measurements being reliable seems to be appropriate, but for the general case, it is not always correct as can be seen in Fig. 2 for an Intel RealSense SR300. The distribution shows a linear relationship between ground truth and measured distance while the color-coding indicates a strong deformation that is influenced by the pixel position within the image. The previous approaches assumed a static deformation, but it can change over time and even between consecutive usages due to external strain or under strong disturbances. The previously described methods are designed for offline processing, making recalibration a costly, time-consuming process while our method can be applied online. We propose an approximation strategy using smooth thin plate splines to acquire pixel-wise correction factors instead of interpolating between multiple pixel bins. Similar to the multi camera calibration case, we use projection to minimize the overall error, but we replace the known calibration target with an arbitrary textured environment without any geometric constraints.

In summary, the key contributions of our paper are:

- depth scaling factors are generated by projection of sparse map points from visual SLAM into depth images,
- the thin plate spline allows for dense approximation of the sparsely distributed correction factors,
- our algorithm can be used online and corrects a RGB-D map in real-time on a modern laptop CPU,
- no one-to-one pixel correspondence is needed, allowing us to use a color camera for calibration,
- evaluation and comparison is done using real world data with and without ground truth.

III. Method

We use an RGB-D extension of the ORB-feature-based SLAM system [13] with keyframes to obtain the depth and color camera trajectory and triangulate a sparse feature map from visual features and corresponding depth measurements. We provide the depth as additional information to the visual SLAM system, but only use visual information for depth calibration. Even though the depth estimate is biased, we prefer to use RGB-D SLAM instead of monocular SLAM since the absolute scale can be fixed and the risk of monocular scale drift is avoided. The correspondence between depth and visual features is established via projection of the measured depth point into the color image with a fixed rigid transformation between color and depth camera. Hence, with a corrected depth estimate the correspondence may change. In case of multiple depth estimates projecting to the same color pixel, we use the one with the shortest range.

The depth calibration is performed asynchronously to the normal tracking and mapping of the system. Since biased depth images may skew the trajectory and map point estimates, we first run a global bundle adjustment (GBA) optimizing the color trajectory and map point positions without the depth measurements. After GBA, the absolute scale may have changed. Hence, we rescale the whole map s.t. the mean ratio of the triangulated and the corresponding measured distance are equal to one. This would obviously not be necessary for a synchronously captured stereo + depth camera as in this case a reliable measure for scale is permanently available.

Subsequently, we estimate the depth correction factors by projecting the sparse triangulated RGB map points into the depth images and compare the measured depth with the triangulated distance relative to the depth camera pose. This is

shown in Fig. 1 where the difference between overestimated measurements (black dashed line) and triangulated distance (gray dotted line) is shown in red. Blue lines denote the inverse case of too near measurements. By also using features without corresponding depth information, we generate a larger sample of scaling factors. Still, the coverage of the scaling factors on the depth sensor is sparse. Hence, we model the depth correction using thin plate splines (TPS) and obtain a smooth correction function from which we generate the pixel-wise depth multipliers. The resulting depth multipliers are then used to rescale the depth images for all depth keyframes. It should be noted that the proposed method is also applicable to a (semi-)dense SLAM system [14], [15], since the visual (inverse-) depth estimate can be easily projected into the depth images. After introducing our notation, we will motivate made assumptions and describe in detail the estimation of the correction factors and the correction model using TPS.

A. Notation

We denote sets and matrices with capital letters and vectors with bold lower case letters. Each map point $\mathbf{p}_w = (x, y, z)^\intercal \in \mathbb{R}^3$ is defined in the world frame w, determined by the initial color camera frame. All poses are represented by a transform $T_{F_2F_1} \in \mathbf{SE}(3)$, which transforms a point $\mathbf{p}_{F_1}$ from the frame F_1 into the frame F_2. For convenience of notation, we identify $T_{F_2F_1}$ with its 4-by-4 matrix. The projection of a point $\mathbf{p}_w$ with pose T_F and camera matrix K_F into frame F gives the image coordinates $\mathbf{u} = (u_x, u_y)^\intercal_F$ in the image domain $\Omega \subset \mathbb{R}^2$ by the mapping:

$$g_F(\mathbf{p}_w) : \mathbf{p}_w \to \mathbf{p}_F, \tag{1}$$

$$(\mathbf{p}_F, 1)^\intercal = T_{Fw} \cdot (\mathbf{p}_w, 1)^\intercal, \tag{2}$$

$$\pi_F(\mathbf{p}_F) : \mathbf{p}_F \to \mathbf{u}_F, \tag{3}$$

$$(x, y, z)^\intercal_F = K_F \cdot \mathbf{p}_F, \tag{4}$$

$$\mathbf{u}_F = (x/z, y/z)^\intercal. \tag{5}$$

In the following, we will use the index c for an RGB camera and d for a depth camera, respectively.

B. Assumptions

We assume that we can obtain the correct depth value by multiplying the measured depth with a scalar factor. This factor may be different for each pixel. Another assumption is local similarity between the correction factors for neighbouring pixels. Teichman et al. previously reported that the range relationship for the Kinect is sublinear and used binning for neighbouring pixels while utilizing multiple scalar factors for the range. Yet, Fig. 2 indicates a pixel depending scalar relationship between ground truth and measured range for a SR300 camera. In case of an RGB-D SLAM system, we further assume the mean of the multiplicative factors to be equal to one, to resolve the scale ambiguity of monocular visual SLAM.

There is a fixed rigid transform $T_{cd} \in \mathbf{SE}(3)$ that transforms a point $\mathbf{p}_d \in \mathbb{R}^3$ from the local coordinate frame of the depth camera into the coordinate frame of the color camera. Hence, we do not need a one-to-one pixel correspondence between depth and RGB image making it possible to obtain a calibration using sensors with different resolutions.

To simplify our approach, we assume corresponding depth and RGB images to be captured synchronously. Even though the assumption of a fixed T_{cd} is in reality often violated due to camera movement during asynchronous capture, incorporating a rigid time-varying transform is straightforward in our approach. Obviously, we will need some texture in the images to extract visual features. Ideally, no part of the scene is occluded, since occluded RGB map points can introduce large depth multipliers and need to be filtered out.

Furthermore, we assume that the intrinsic camera matrices K_c, K_d have been calibrated for the color and depth camera and image distortion is removed beforehand.

C. Depth Correction Factor Estimation

After performing SLAM, a global bundle adjustment refines the color camera poses T_{cw} and triangulated map point positions $\mathbf{p}_w$ based on visual information only. If we would incorporate depth measurements, the error induced by the deformation will be distributed between map, poses and depth calibration. We assume the map to have absolute scale, which is true for stereo cameras, but not for a monocular camera. Hence, we scale the refined $\mathbf{p}_w$ and the translation of T_{cw} such that the mean factor between all measured distances and all triangulated distances is equal to one.

The basic idea to obtain pairs of expected and measured depth, as visualized in Fig. 1, is simply to project map points into the depth image and compare the triangulated distance z_t: $(x, y, z_t)^\intercal_d = \mathbf{p}_d = g_d(\mathbf{p}_w)$ with $T_{dw} = T_{dc} \cdot T_{cw}$, (6)

with the measured depth z_m in the depth image at $\pi_d(\mathbf{p}_d)$.

This allows us to calculate a scaling factor $s_\mathbf{u} = \frac{z_t}{z_m}$ which is one in the ideal case. A factor greater than one means the depth is measured too short; smaller than one means too far. Yet, both variables are affected by noise which has to be taken into account. Assuming Gaussian white noise, we can rewrite the scaling as follows [16]:

$$z_t = s_\mathbf{u} \cdot z_m \text{ with } z_m \sim \mathcal{N}\left(s_\mathbf{u}\mu, \sigma_m^2\right), z_t \sim \mathcal{N}\left(\mu, \sigma_t^2\right). \tag{7}$$

The maximum likelihood (ML) estimator to this scaling problem was shown by Engel et al. [16] to have a unique solution. Since we are not interested in μ, we only estimate the scale parameter. We cannot rely on gathering a high number of scale estimates per pixel if we want to run our method online. Having multiple factors for one pixel rarely happens for sparsely distributed map points. Hence, the thin plate spline is a convenient method to interpolate between correction factors.

D. Depth Correction Model

We use thin plate splines (TPS) to model the correction factors w.r.t. the depth image coordinates. Due to the excellent fill-in property and the minimal bending energy of these splines, this works in general even with scattered, sparsely

distributed data — in our case the correction factors and corresponding image position — while giving smooth function approximations with a small number of coefficients. Here, we use the following two-dimensional thin plate polyharmonic spline:

$$f(\mathbf{u}) = \sum_{i=1}^{N} c_i \cdot \phi\left(|\mathbf{u} - \mathbf{d}_i|\right) + \mathbf{v}^\intercal \cdot \begin{pmatrix} 1 \\ \mathbf{u} \end{pmatrix}, \tag{8}$$

with the radial basis function (RBF):

$$\phi(r) = r^2 \cdot \ln(r). \tag{9}$$

Here, $\mathbf{u}$ is the data point, in our case a pixel coordinate, and $\mathbf{d}_i \in \Omega$ is a control point within the image. The parameters $\mathbf{c}$ control the influence of the RBF, while $\mathbf{v}$ is a polynomial which aids the approximation. One advantage of the TPS is the lack of parameters that have to be tuned, since $\mathbf{c}, \mathbf{v}$ are calculated from the given image positions $\mathbf{u}$ and the desired function values, the correction factors $s_\mathbf{u}$. Furthermore, TPS is far more flexible compared to a polynomial with the same number of coefficients.

In case of interpolation, one seeks to find the coefficients $[\mathbf{c}, \mathbf{v}]^\intercal$ s.t. the following equations are satisfied:

$$s_i = f(\mathbf{u}_i), 1 \leq i \leq M. \tag{10}$$

Since the interpolation would require as many RBFs (N) as there are data points (M), this cannot be used efficiently online. Instead, we approximate the underlying function using a grid with a small fixed number of $N = L \times L$ control points:

$$\arg\min_{\mathbf{c},\mathbf{v}} \sum_{i}^{M} \left\| f(\mathbf{u}_i) - s_i \right\|^2. \tag{11}$$

On each control point $\mathbf{d}_i$, one RBF will be statically placed. We typically choose $L \in \{3, 4, 5\}$, but other choices and different grids are possible as well.

Since we have $N+3$ unknown variables for $[\mathbf{c}, \mathbf{v}]^\intercal$, often the following three additional conditions are added:

$$\sum_{i}^{N} c_i = 0, \sum_{i}^{N} c_{i,x} \cdot d_{i,x} = 0, \sum_{i}^{N} c_{i,y} \cdot d_{i,y} = 0, \tag{12}$$

which ensure that a polynomial of maximal degree two can be correctly approximated. Higher-order and local deformations are caught by the RBFs.

To minimize the least squares error function Eq. (11) such that Eq. (12) is satisfied, we have to solve the augmented system of equations, stated in matrix form as:

$$\begin{pmatrix} A & X \\ D & 0 \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{s} \\ \mathbf{0} \end{pmatrix}, \tag{13}$$

$$A = \begin{bmatrix} \phi(|\mathbf{u}_1 - \mathbf{d}_1|) & \dots & \phi(|\mathbf{u}_1 - \mathbf{d}_N|) \\ \vdots & \ddots & \vdots \\ \phi(|\mathbf{u}_M - \mathbf{d}_1|) & \dots & \phi(|\mathbf{u}_M - \mathbf{d}_N|) \end{bmatrix},$$

$$X = \begin{pmatrix} 1 & \mathbf{u}_1 \\ \vdots & \vdots \\ 1 & \mathbf{u}_M \end{pmatrix}, D = \begin{pmatrix} 1 & \dots & 1 \\ \mathbf{d}_1 & \dots & \mathbf{d}_N \end{pmatrix}.$$

Since we have $M \gg N$, this can be solved like any over-determined system. We decided to employ conjugate gradient (CG), since CG allows to use an initial solution, whereas a QR-decomposition does not. Hence, we can speed up the calculation in an online fashion using the previous estimate. This further enables us to use iterative reweighted least squares (IRLS) [17] for outlier rejection, if we have sufficient data points.

E. Range Dependent Extension

An extension of the approach to range dependent scaling factors could be to estimate multiple TPS for different depth range intervals, similar to [5]. A more straightforward extension is to include the measured depth z_m into $f(\mathbf{u})$ by augmenting $\mathbf{u}$ with z_m as $\mathbf{u}_a = (u_x, u_y, z_m)^\intercal \in \Omega \times \mathbb{R}$ and using control points $\mathbf{d}_a \in \Omega \times \mathbb{R}$. This can be seen as placing multiple RBFs on a pixel at different depths. Hence, N increases, e.g., to $L \times L \times L$. The range-dependent formulation is again minimized using (13) and similarly evaluated with (11), both using the augmented $\mathbf{u}_a$ and $\mathbf{d}_a$ instead of $\mathbf{u}$ and $\mathbf{d}$. Unfortunately, the evaluation of $f(\mathbf{u}_a)$ cannot as effectively be precomputed as in the two-dimensional case, since it needs to be computed for every depth measurement.

IV. Implementation

Due to the properties of the projection, the aforementioned association between depth measurements and visual features may change after rescaling the depth images. Hence, iteration of the calibration process is needed. In our case, we asynchronously run the calibration after keyframe creation.

The most time-consuming operation is the application of the calibration by rescaling the depth image and reprojection into the color image to obtain the association between visual features and depth measurement. Therefore, we use vectorization and Eigen library expressions [18] as often as possible and minimize the intermediate evaluation calls. Instead of calculating Eq. (6) for each point separately, we first calculate the projection matrix from π_{cw} and then multiply it with the column-wise stacked $3 \times M$ matrix containing all M map points. Similarly, if we assume synchronized capture between depth and RGB camera, we can precompute the concatenated backprojection π_d^{-1} and projection π_c for each pixel as underlined:

$$\pi_{cd}(\mathbf{u}) = \underline{K_c \cdot R_{cd} \cdot K_d^{-1} \cdot (\mathbf{u}, 1)^\intercal} \cdot z_m + \underline{K_c \cdot \mathbf{t}_{cd}}. \tag{14}$$

Hence, at runtime we only need to evaluate a scalar-vector-product and a vector-vector-addition per pixel, reducing the overall computation time.

To further increase the accuracy of the SLAM system, we continuously search additional correspondences between keyframes that belong to the same local map. This process is similar to the local loop closures in ElasticFusion [2]. The reasoning behind this is that pose error induced by miscalibrated depth estimates or small tracking errors may discard correct correspondences between keyframes and hence, degrade the overall performance and integrity of the RGB-D map. After calibration, these correspondences can be re-established.

Images typically underlie distortions that have to be taken into account for reconstruction. To address this, we undistort the depth image using OpenCV with the radial and tangential distortion parameters from either factory calibration or Kalibr, and solely work on the undistorted depth. After projecting map points into the depth camera, we apply the distortion and calculate the depth correction factors at the distorted image positions. In order to generate the pixel-wise depth correction factors for the undistorted depth image, we evaluate the TPS at the distorted image positions that correspond to the undistorted pixels.

Furthermore, it should be noted that due to the convergence of the calibration solution, rescaling for a keyframe is done only sporadically, e.g. after ten calibration runs if pixel-wise depth correction factors between two consecutive calibrations do not change too much.

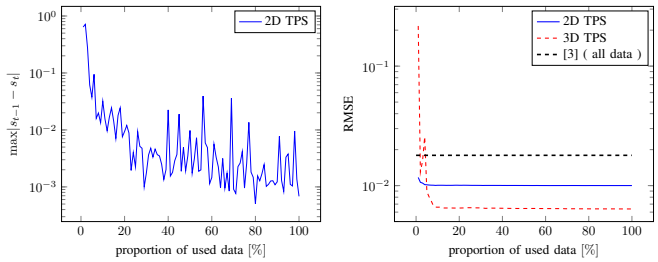
V. Evaluation

We wanted to compare our calibration method against the methods by di Cicco et al. (EDC) and Teichman et al. (CLAMS) with focus on the SR300, but also for the popular Kinect, Xtion and PrimeSense RGB-D sensors. Unfortunately, an evaluation of our approach on the dataset by di Cicco et al. [3] for Kinect and Xtion was not possible due to missing color images. Instead, we tested the algorithms for the Xtion on the widely used TUM dataset [19] using 'freiburg3_long_office_household'. We further captured a trajectory within a hallway using a PrimeSense Carmine.

Additional experiments were undertaken for the SR300. We captured data for three different scenarios: aprilgrid, boxes and walls. For the aprilgrid dataset, an planar grid of AprilTags [20] is placed on a table. The grid allows to estimate the sensor movement, while the geometry simplifies the comparison between corrected and expected measurements. For the boxes dataset, three large boxes were placed parallel to each other. The outer boxes were tilted inwards and all are filled with various objects to generate a non-planar and more challenging environment with occlusions. Ground truth depth was acquired by a Photoneo PhoXi 3D Scanner XL with millimeter accuracy. For the walls dataset, the SR300 covered several non-planar stonework surfaces in rapid motion. We found the same basic distortion pattern on multiple tested SR300, hence we used the same SR300 for all our datasets. Each image stream is captured at 30 fps with 640×480, except for color which has resolution 1920×1080. Since the camera is moved quickly, the exposure time for color is fixed to 8 ms to reduce motion blur, which was severe using auto exposure even in well illuminated scenes.

For all datasets we use the factory calibration reported by each device for the intrinsics, if not otherwise stated. All experiments are run at least five times. The respective errors are than averaged over all tries. All calculations were performed on an Intel Core i7-6700 HQ with 32GB RAM.

We start depth calibration after ten keyframes have been created and placed the control points on a 5×7 regular grid to incorporate the aspect ratio. IRLS is iterated up to 100 times or until the difference between consecutive iterations



(a) convergence of the pixelwise scaling factor

(b) RMSE after subsequent calibrations

Fig. 4: Convergence behaviour of our method for increasing proportions of data on the Aprilgrid dataset.

is below 0.001. The weights are obtained using the absolute deviation between estimated and measured correction factor with regularization at 0.0001 to avoid division by zero. Due to realtime constraints, we do not use the 3D extension from Sec. III-E, since the evaluation for the whole image is too computationally involved.

A. Xtion and PrimeSense Carmine

No ground truth map data is provided in the TUM dataset, only camera ground truth poses from a motion capture system are available. Hence, we evaluate the Absolute Trajectory Error (ATE) [19] with and without depth calibration. The results are summarized in Table I. EDC was not evaluated, since no dataset provided sufficient planar surfaces to estimate a good calibration.Even though the smoothness assumption might not hold for this type of sensor, due to the flexibility of the TPS the results do not degrade. Yet, the small reduction of the ATE from 3.3 cm down to 2.3 cm is not necessarily due to the depth calibration, but may stem from the frequent application of GBA. For our hallway dataset we use the mean map entropy (MME) [21] to assess the reconstruction quality, since a smaller MME corresponds to a visually sharper point cloud.

B. Intel Realsense SR300

The correction model for EDC was created by moving the SR300 facing the floor from 2 m slowly towards 0.5 m height. While the method worked well on all SR300 datasets, the estimated correction by CLAMS was increasing the deformation and hence immediately diverged. This happened likely due to the strong deformation of the map, as visualized in Fig. 5 for the wall dataset, and since the actual measured data violates the assumption of being approximately correct at 1 m. Their underlying SLAM system strongly relies on the measured depth. The result did not improve when only rgb was used for updating the poses during tracking. Hence, we will not report results for CLAMS on the SR300 datasets.

In order to evaluate the performance of [3] and our method on the aprilgrid dataset, we compute the RMSE between ground truth distance z_{gt} and scaled depth measurement $s_{\mathbf{u}} \cdot z_m$ as

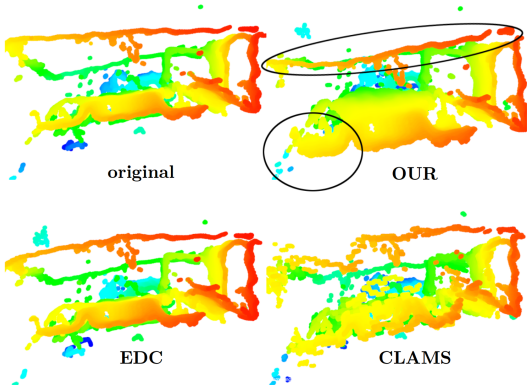


Fig. 5: Point cloud from the boxes dataset, color coded from close (red) to far (blue). The top ellipse shows the matching edges of the box. The left and right side have the same height. EDC does not correct enough. CLAMS pulls the box apart.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(z_{gt} - s_{\mathbf{u}} \cdot z_m)^2}. \tag{15}$$

The convergence of the RMSE can be seen in Fig. 4b. To obtain a lower bound on the RMSE, we also fitted the splines on increasing portions randomly taken from the whole image sequence. To show the convergence of the proposed method we compute the maximum difference

$$\max_{\mathbf{u}\in\Omega} |s_{t-1,\mathbf{u}} - s_{t,\mathbf{u}}| \tag{16}$$

in the multiplier map s_t between subsequent calibration runs at time t and $t-1$, respectively. As can be seen in Fig. 4b, the RMSE of our method after using one percent of the data is already lower than that of EDC (dashed black line) and our method converges very quickly. The range-dependent 3D-TPS extension (cfg. Sec. III-E) can further reduce the RMSE (red dashed line) while it takes longer to converge. This convergence can also be seen in Fig. 4a for the pixel-wise correction factors as the maximum difference between subsequent calibrations decreases. Hence, we could start the calibration process online and stop after convergence while continuing SLAM.

For the boxes dataset we manually preregister the calibrated point clouds against the ground truth cloud, apply ICP, and visually verify the result. To estimate how well the cloud matches, we calculate the mean distance to the nearest neighbor (NNE). Fig. 6 shows the color coded error w.r.t. ground truth. Red areas with larger error stem especially from regions without any depth measurements due to occlusion as can be seen on the upper block or on the sides of the boxes. Fig. 7 shows for the wall dataset the estimated aggregated point cloud of our SLAM system and the corresponding scaling over time[1]. The error reduction on the straight wall is clearly visible as the bending is reduced quickly. The depth correction factors swiftly become stable.

[1]An accompanying video is available at `https://www.ais.uni-bonn.de/videos/IROS_2017_depth_calibration`.

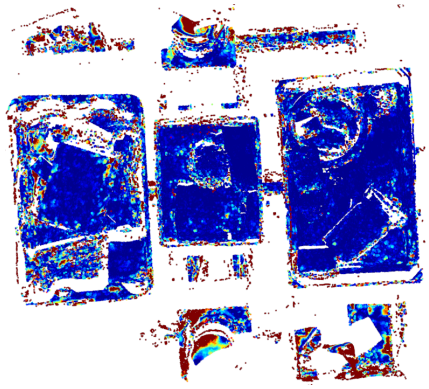


Fig. 6: Distance to ground truth for the box dataset using our calibration. Coloring encodes distance from low (blue) to high (red).

TABLE I: Error comparison for the tested datasets

Error	Dataset	none	CLAMS [5]	EDC [3]	ours
ATE [cm]	fr3office	3.350	4.460	-	2.339
MME	hallway	-2.548	-1.585	-2.501	-2.639
	boxes	-7.817	-7.542	-7.884	-7.996
	walls	-5.658	-	-6.068	-6.326
NNE [cm]	boxes	0.391	0.590	0.276	0.263

C. Runtime

The precomputed and optimized projection is faster by a factor of 5.17 than the non optimized projection (85 ms). The additional correspondence search found on average 51 correspondences, while taking approx. 10.14 ms. The regular number of matched map points is in the scope of 300.

The overall calibration time took on average less than 0.5 s, but increases with the number of keyframes and map points. The vast majority of time is spent on solving Eq. (13) with IRLS for outlier rejection. Hence, fewer iterations could reduce computation time, but we have seen in our experiments that using the Least Squares Solution can skew the calibration especially at the beginning and may result in tracking failure.

VI. Conclusions

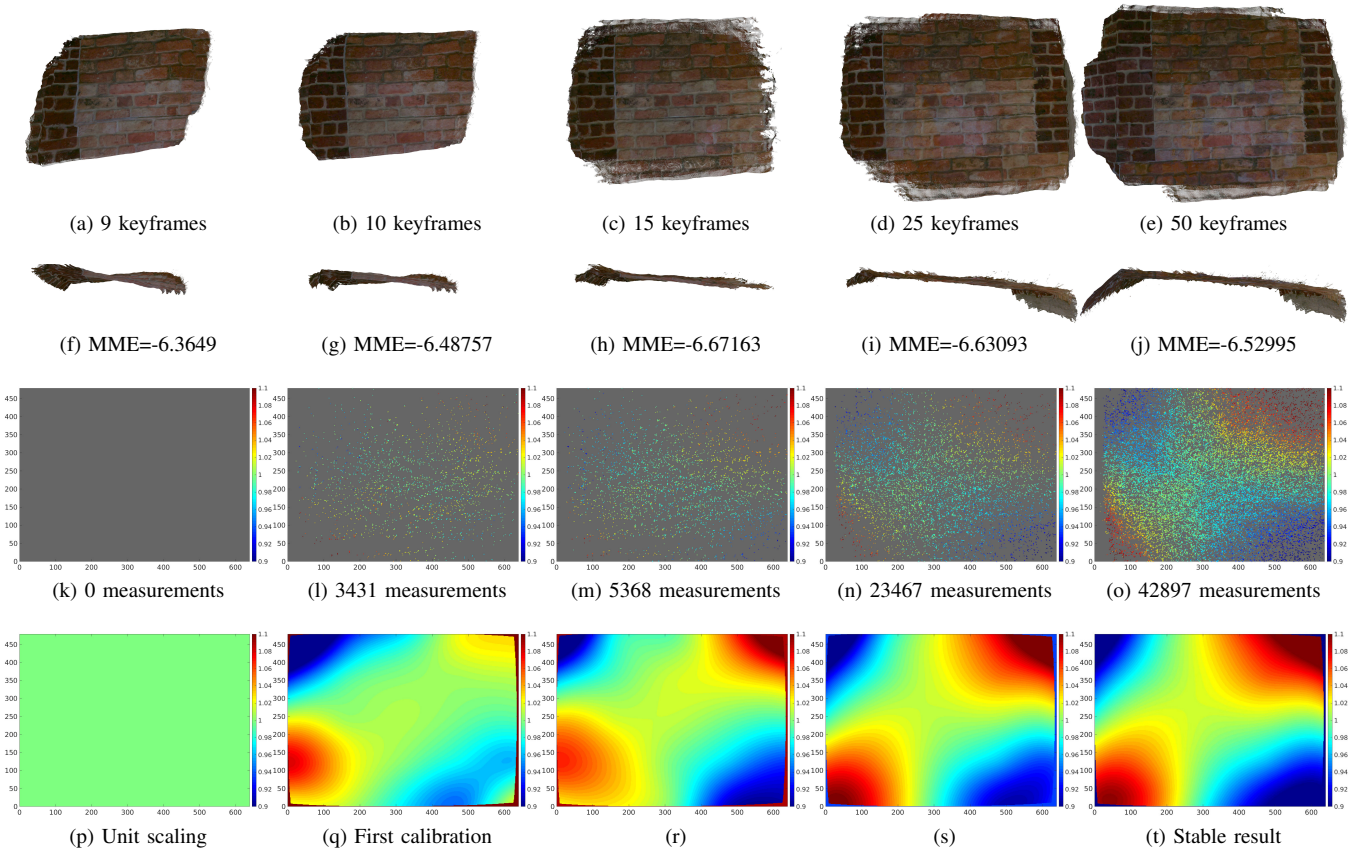
We presented a fast and simple depth calibration method using a visual SLAM system running online without requiring planar surfaces, calibration targets, or known map geometry. We employ thin plate splines for approximating the depth correction factors w.r.t. the image position to deal with sparsely distributed correction estimates. The experimental results substantiate that the calibration converges quickly and effectively corrects depth deformations.

Acknowledgment

This work was supported by grants BE 2556/7 and BE 2556/8 of the German Research Foundation (DFG) and 608849 of the European Union's 7th FP. We would like to thank Photoneo for providing the PhoXi 3D Scanner XL.

References

[1] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: real-time dense surface mapping and tracking," in *Proc. of the IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.



(a) 9 keyframes (b) 10 keyframes (c) 15 keyframes (d) 25 keyframes (e) 50 keyframes

(f) MME=-6.3649 (g) MME=-6.48757 (h) MME=-6.67163 (i) MME=-6.63093 (j) MME=-6.52995

(k) 0 measurements (l) 3431 measurements (m) 5368 measurements (n) 23467 measurements (o) 42897 measurements

(p) Unit scaling (q) First calibration (r) (s) (t) Stable result

Fig. 7: Sequence of wall dataset, the columns present the results after creation of 9, 10, 15, 25 and 50 keyframes. The first calibration starts after 10 keyframes are created. Top two rows: aggregated point clouds in frontal and top view. Middle Row: sparsely distributed measurements, color-coded by depth correction factor. Bottom Row: estimated undistorted dense correction factors. For this visualization was a manual intrinsic calibration used.

[2] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: dense SLAM without a pose graph." in *Proc. of Robotics: Science and Systems*, 2015.

[3] M. Di Cicco, L. Iocchi, and G. Grisetti, "Non-parametric calibration for depth sensors," in *Proc. of the Int. Conference on Intelligent Autonomous Systems (IAS)*, 2014.

[4] F. Basso, A. Pretto, and E. Menegatti, "Unsupervised intrinsic and extrinsic calibration of a camera-depth sensor couple," in *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, 2014.

[5] A. Teichman, S. Miller, and S. Thrun, "Unsupervised intrinsic calibration of depth sensors via SLAM," in *Proc. of Robotics: Science and Systems*, 2013.

[6] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2013.

[7] D. Herrera, J. Kannala, and J. Heikkilä, "Joint depth and color camera calibration with distortion correction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[8] J. Smisek, M. Jancosek, and T. Pajdla, "3D with Kinect," in *Consumer depth cameras for computer vision*, 2013.

[9] C. Nguyen, S. Izadi, and D. Lovell, "Modeling Kinect sensor noise for improved 3D reconstruction and tracking," in *Proc. of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, 2012.

[10] S. Fuchs and G. Hirzinger, "Extrinsic and depth calibration of ToF-cameras," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[11] D. Ferstl, C. Reinbacher, G. Riegler, M. Rüther, and H. Bischof, "Learning depth calibration of time-of-flight cameras," in *Proc. of the British Machine Vision Conference (BMVC)*, 2015.

[12] S. Clarkson, J. Wheat, B. Heller, J. Webster, and S. Choppin, "Distortion correction of depth data from consumer depth cameras," *Proc. of the International Conference on 3D Body Scanning Technologies*, 2013.

[13] S. Houben, J. Quenzel, N. Krombach, and S. Behnke, "Efficient multi-camera visual-inertial SLAM for micro aerial vehicles," in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2016.

[14] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2014.

[15] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *arXiv preprint arXiv:1607.02565*, 2016.

[16] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2012.

[17] P. Holland and R. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics-theory and Methods*, 1977.

[18] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[19] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the evaluation of RGB-D SLAM systems," in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2012.

[20] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, 2011.

[21] J. Razlaw, D. Droeschel, D. Holz, and S. Behnke, "Evaluation of registration methods for sparse 3D laser scans," in *Proc. of the European Conference on Mobile Robots (ECMR)*, 2015.