Accurate Intrinsic and Extrinsic Calibration of RGB-D Cameras with GP-based Depth Correction

Guangda Chen, Guowei Cui, Zhongxiao Jin, Feng Wu and Xiaoping Chen

***Abstract*—In recent years, more and more robots have been equipped with low-cost RGB-D sensors, such as Microsoft Kinect and Intel Realsense, for safe navigation and active interaction with objects and people. In order to obtain more accurate and reliable fused color and depth information (coloured point clouds), not only the intrinsic and extrinsic parameters of color and depth sensor should be precisely calibrated, but also the external corrections of depth measurements are required. In this paper, using motion capture system, we propose a reliable calibration framework that enables the precise estimation of the intrinsic and extrinsic parameters of RGB-D sensors and provide a model-free depth calibration method based on heteroscedastic Gaussian Processes. Compared with the existing depth correction techniques, our method can simultaneously estimate the mean and variance of the depth error at different measurement distances, i.e., the probability distribution of the depth error relative to the measured distance, which is essential in the state estimation problems. To verify the effectiveness of our approach, we conduct a thorough qualitative and quantitative analysis of the major steps of our calibration method, and compare our experimental results with other related work. Furthermore, we demonstrate an experiment about the overall improvement of visual SLAM with a Kinect device calibrated by our calibration technique.**

***Index Terms*—RGB-D cameras, calibration, motion capture system.**

I. Introduction

AS MANY domestic robot tasks, such as visual SLAM [1], robot navigation [2], object recognition [3] and manipulation [4], require fused color and depth information, more and more service robots are equipped with color (RGB) and depth (D) cameras. Basically such RGB-D sensors consist of a depth camera rigidly attached to a color camera, where the depth sensor can be a time-of-flight (ToF) camera or a sensor based on structured light. In this paper, we focus on Kinect-like sensors (structured light sensors, like Intel Realsense SR300 and Asus Xtion Pro Live) because they are widely used in robotics. Although these devices are factory calibrated, the quality of this calibration is adequate mostly for video games. Therefore, we still need a proper calibration procedure for robust robotics applications. For this kind of sensors, we must calibrate the intrinsic parameters (focal length, principal point, and lens distortion) of both the color and depth camera and space alignment (extrinsic parameters, i.e., relative position and orientation) between them. Moreover, we have noticed that for longer distances, there is an increasing deviation and uncertainty in depth measurements and the variation of measurement bias and uncertainty is not stationary in different pixel coordinates. Therefore, the complete calibration of RGB-D cameras is mainly divided into the determination of intrinsic and extrinsic parameters and the correction of depth measurements.

In order to obtain intrinsic and extrinsic parameters of the depth camera, there are various calibration methods based on different sources of depth camera, e.g., infra-red (IR) images, disparity images or depth images. Early works, such as [5], were based on IR images and the standard RGB camera calibration techniques. They use a checkerboard pattern to calibrate the intrinsic parameters of RGB and IR cameras and also the extrinsic transformation between them. However, their method requires different cameras to observe the same checkerboard pattern at the same time and becomes cumbersome and imprecise because the most widely used ROS Kinect driver [6] does not support simultaneous output of IR and RGB images. The work of Herrera *et al.* [7] is a typical method based on disparity images. They let the kinect view a large plane attached with a checkerboard at different distances. There are two major weaknesses of their work: Firstly, they require an expert user to play an active roll in selecting corresponding features in the depth images. Secondly, their method relies heavily on specific Kinect-type disparity data. Other methods, based on the depth images, make use of specific calibration objects simultaneously observed by the RGB and depth sensors. In [8], the authors use a large custom-made wood panel with tens of circular holes and it requires an extensive manual procedure to associate the holes' centers in each RGB and depth image pair. Moreover, in [9], the authors use a spherical basketball. The common limitation of the depth image based method is that the noise of the depth map is very large and the result is not accurate enough.

For depth correction, Smisek *et al.* [10] showed that Kinect sensors are affected by some radially symmetric distortions, and then Herrera *et al.* [7] gave a first attempt to take into account of the distortion in their calibration process. However, their method uses an external high-definition camera to observe the checkerboard and is not suitable for long distances between RGB camera and depth sensor. Zhang *et al.* [11] showed that the depth measurement provided by Kinect was a linear function of the real depth. The main problem of depth correction is how to accurately obtain the true depth of each

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1002204, in part by the National Natural Science Foundation of China under Grant U1613216 and Grant 61603368, and in part by the Youth Innovation Promotion Association of CAS under Grant 2015373. *(Corresponding author: Xiaoping Chen.)*

The authors are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: cgdsss@mail.ustc.edu.cn; cuigw@mail.ustc.edu.cn; zxjin@mail.ustc.edu.cn; wufeng02@ustc.edu.cn; xpchen@ustc.edu.cn).

Digital Object Identifier 10.1109/JSEN.2018.2889805

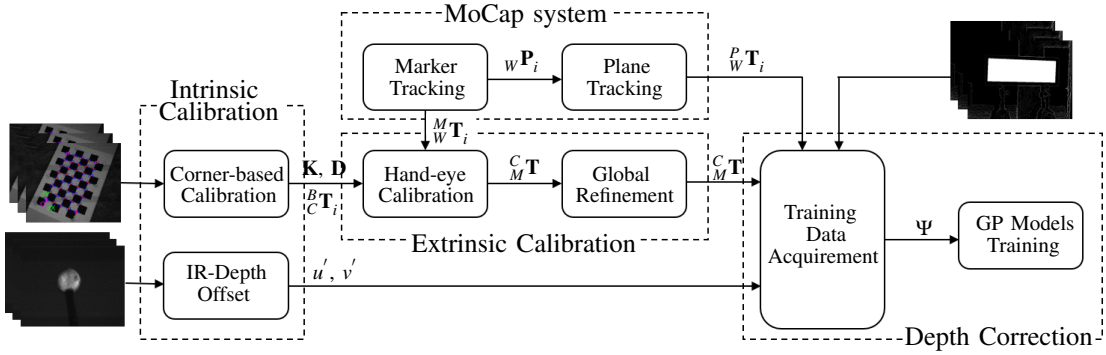


Fig. 1: Work-flow diagram of the RGB-D camera calibration approach.

pixel in the depth images. Teichman *et al.* [12] assumed the close-up point cloud maps, generated by SLAM previously, can be used as the true values of depth measurements in the corresponding position. Obviously their method heavily relies on the positioning and mapping accuracy of SLAM. Most recently, Basso *et al.* [13] separated the depth error into two different parts: *distortion error* and *global error*, and treated them separately. They assume that there is no error in the nearest depth measurement and depth errors are the same for the near depth measurements in the same pixel location. Because they use a polynomial curve fit to correct the depth value, the polynomial form needs to be determined in advance.

Inspired by the work [14] that provided a very popular benchmark for the evaluation of RGB-D SLAM, we propose a novel calibration approach that removes the systematic hypothesis of acquisition for groud truth of depth measurements and is based on the calibration framework [15] which depends on high-precision measurement of the motion capture system (MoCap). Different from the extrinsic parameters calibration method proposed in [14], our method, based on hand-eye calibration techniques, eliminates the inevitable error caused by inaccurate position of the markers attached on the checkerboard. Here, we still make use of IR images, mainly because of the characteristics of low noise compared to the depth map, to estimate the intrinsic and extrinsic parameters. Compared with the traditional IR-based methods, our approach does not directly estimate the spatial relation between the depth camera and the color camera, which requires the depth camera and the color camera to observe the same checkerboard at the same time, but to calibrate the spatial transformation relationships between the camera frames and the global world frame provided by motion capture system. The additional benefit of this method is that we can easily scale to calibrate extrinsic parameters between multiple cameras [16] [17] or with other sensors [18] [19]. For the depth measurements provided by the depth sensor, we employ an error model that can reduce the distortion and systematic errors and get more accurate depth measurements of the environment. This model is represented as a set of model-free Gaussian Processes (GP). To obtain the model parameters, we make full use of the motion capture system to get the corresponding ground truth and measured depth of each pixel at different distances. More importantly, we found that as the measured distance grows, the uncertainty of the measured depth value also increases, i.e., the variance of the error increases. Hence we adopt sparse heteroscedastic Gaussian processes [20] to estimate both the mean and variance of the measurement error, i.e., the probability distribution of the depth error relative to the measured distance, which is essential in the state estimation problems in robotics research.

Our main contributions are summarized as follow:

- A complete and accurate calibration protocol of Kinect-like sensors, which includes intrinsic, extrinsic calibration and depth correction.
- A joint intrinsic and extrinsic calibration method of cameras, which is mainly based on the customized hand-eye calibration techniques.
- Direct and non-recursive data acquirement for the ground truth of pixel-wise depth measurements and a model-free GP-based depth correction method for estimating both the mean and variance of depth errors.

A. Problem definition and approach overview

From a pair of depth image τ_D and color image τ_R, the method for recovering the coloured 3D information is as follows.

For ith pixel (u_D^i, v_D^i) in depth image τ_D, the depth measurement is represented as $z(u_D^i, v_D^i)$ and the corresponding pixel in IR image is (u_I^i, v_I^i). Then the 3D pose $\mathbf{M}_I$ of pixel (u_I^i, v_I^i) can be recovered by the camera imaging model with depth measurement. That is

$$\overset{o}{\mathbf{M}}_I = project(\mathbf{K}_I, \mathbf{D}_I, u_I^i, v_I^i) \tag{1}$$

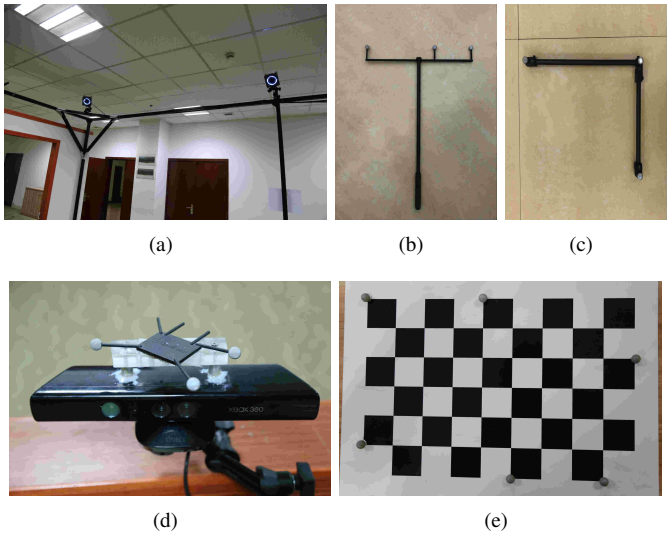
$$\mathbf{M}_I = \overset{o}{\mathbf{M}}_I \cdot z(u_D^i, v_D^i) \tag{2}$$

where $\mathbf{K}_I$ is the intrinsic parameters of IR camera and $\mathbf{D}_I$ is the distortion parameters, $\overset{o}{\mathbf{M}}_I$ is normalized form ($z = 1$) of $\mathbf{M}_I$. After that, we can transform pose $\mathbf{M}_I$ in IR frame to $\mathbf{M}_R$ in RGB frame by using the transformation ${}^R_I\mathbf{T}$ from IR frame to RGB frame. That is

$$\mathbf{M}_R = {}^R_I\mathbf{T} \cdot \mathbf{M}_I \tag{3}$$

Then, we can re-project $\mathbf{M}_R$ to pixel (u_R^i, v_R^i) in RGB image τ_R and get the color information $c(u_R^i, v_R^i))$ in the color image τ_R:

$$(u_R^i, v_R^i) = repr(\mathbf{K}_R, \mathbf{D}_R, \mathbf{M}_R) \tag{4}$$



(a) (b) (c)

(d) (e)

Fig. 2: (a) Our motion capture system. (b) and (c) are calibration tools of MoCap system. (d) The Kinect device which rigidly attached with three markers. (e) the checkerboard pattern used for intrinsic and extrinsic calibration and also for evaluation of our calibration when together with attached six markers.

where $\mathbf{K}_R$ and $\mathbf{D}_R$ are the intrinsic parameters and distortion parameters of RGB camera, $repr()$ is the reverse process of $project()$.

The goal of our calibration approach is to enable the RGB-D cameras to recover more accurate coloured 3D environmental information from RGB and depth image pairs, which means more accurate depth information and better alignment between color and depth measurements of all coloured 3D points $(\mathbf{M}_R, c(u_R^i, v_R^i))$. As shown in Fig. 1, our work mainly consists of three parts: intrinsic calibration, extrinsic calibration and depth correction. Firstly, in Section II-A, we calibrate the intrinsic parameters of IR and RGB camera which include $\mathbf{K}_I$, $\mathbf{D}_I$ and $\mathbf{K}_R$, $\mathbf{D}_R$, and the relationship between corresponding pixel coordinates in IR image and depth image can be obtained with the method mentioned in Section II-B. Meanwhile, the extrinsic parameters ${}_I^R\mathbf{T}$ are obtained in Section III. Finally, we provide the method to get more accurate depth measurements $z(u_D^i, v_D^i)$ in Section IV.

II. Intrinsic Calibration

The RGB and IR camera both commonly follow a so-called pinhole camera model with radial distortion and tangential distortion. That is, the relationship between a 3D point $\mathbf{M} = [X, Y, Z, 1]^T$ and its pixel coordinate of image projection $\mathbf{m} = [u, v, 1]^T$ is given by

$$s\mathbf{m} = \mathbf{K}[\mathbf{R}, \mathbf{t}]\mathbf{M} \tag{5}$$

where s is a scale factor, $\mathbf{R}$ and $\mathbf{t}$ is the rotation and translation respectively which relates some other coordinate system to the camera coordinate system and together are called the extrinsic parameters of the camera, and the intrinsic matrix $\mathbf{K}$ is given by

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y are the focal lengths in image x and y axes, γ is the parameter which describe the skewness of the two axes, and (u_0, v_0) is the coordinates of principal point.

Besides, real lenses usually have some distortion, mostly radial distortion and slight tangential distortion. the relationship between ideal image coordinate (x_n, y_n) and real (distorted) image coordinate (x_t, y_t) is given by

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} x_n(1 + k_1 r^2 + k_2 r^4) \\ y_n(1 + k_1 r^2 + k_2 r^4) \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} 2p_1 x_r y_r + p_2(r^2 + 2x_r{}^2) \\ p_1(r^2 + 2y^2) + 2p_2 x_r y_r \end{bmatrix} \tag{7}$$

where $r^2 = x_n^2 + y_n^2$, k_1 and k_2 are radial distortion coefficients, as well as p_1 and p_2 are tangential distortion coefficients.

A. Corner-based calibration and target extract

The intrinsic calibration of cameras is a well-studied problem. In Zhang's method [21], he assumed the model plane is on Z = 0 of the world coordinate system without loss of generality. The checkerboard corners are extracted from the intensity image. And then each homography is computed for each image using the known corner position in the world coordinates and measured positions in the image. A linear system of equations is used to solve the constraints imposed by each homography. After that, the complete set of parameters can be estimated by minimizing the following function:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \left\| \mathbf{m}_{ij} - \breve{\mathbf{m}}(\mathbf{K}, k_1, k_2, p_1, p_2, \mathbf{M}_{ij}, {}_C^B\mathbf{T}_i) \right\|^2 \tag{8}$$

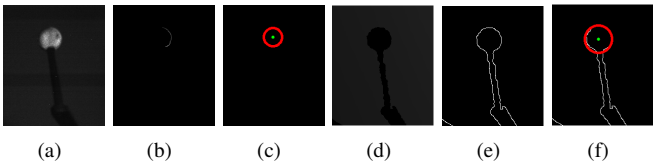
where the function $\breve{\mathbf{m}}$ is the projection of jth point $\mathbf{M}_{ij}$ in image i according to equation (5), followed by radial distortion (equation (6)) and tangential distortion (equation (7)). This nonlinear minimization is solved by the Levenberg-Marquardt Algorithm. An initial value of ${}_C^B\mathbf{T}_i$ (from camera frame C to checkerboard frame B) which contains a translation and a rotation in the Rodrigues formula, and intrinsic parameters $\mathbf{K}$ are obtained using the previous estimations. the distortion parameters k_1, k_2, p_1, p_2 are initialized with the technique described in Zhang's method [2], or just set to be 0. By doing this calibration, we can not only obtain the intrinsic parameters, but also get all the poses of the camera in the checkerboard coordinate system, ${}_C^B\mathbf{T}_i$, which are used for the extrinsic calibration described in the next section.

B. Offset between IR image and depth image

As reported in [22], there is a fixed offset between corresponding pixels in IR image and depth image. That is

$$\mathbf{m}_I = \mathbf{m}_D + \begin{bmatrix} u' & v' & 0 \end{bmatrix}^T \tag{9}$$

In order to estimate the value, we point the camera towards the marker of our calibration tool as shown in Fig. 2(b) or Fig.



(a) (b) (c) (d) (e) (f)

Fig. 3: IR (a-c) and depth (d-f) images during different processing: (a) Original IR image. (b) After Canny edge detection. (c) Detected circle by Hough circle detection. (d) Original depth image. (e)After Canny edge detection. (f) Detected circle by Hough circle detection.

2(c) and keep the marker still. Then 180 pairs of IR and depth images are recorded. For each image pair, we first use Canny edge detection [23] and then Hough circle detection [24] to detect the center point of the marker in IR images (some actual effects are shown in Fig. 3(a),3(b), 3(c)) and depth images (as shown in Fig. 3(d), 3(e), 3(f)). Due to the depth image of Kinect is very noisy, we can hardly get accurate results of u', v'. The average estimated value of (u', v') we got in our experiments is about (4, 4).

III. Extrinsic Calibration

During the intrinsic calibration, we move the camera to the different poses in front of the checkerboard pattern which remains motionless. As shown in Fig. 4, the coordinate system of the camera optical center moves from C_1 to C_2 and at the same time the markers coordinate system transforms from M_1 to M_2 correspondingly. So we have

$$ {}^{B}_{W}\mathbf{T} = {}^{C_1}_{B}\mathbf{T}^{-1} \cdot {}^{C_1}_{M_1}\mathbf{T} \cdot {}^{M_1}_{W}\mathbf{T} = {}^{C_2}_{B}\mathbf{T}^{-1} \cdot {}^{C_2}_{M_2}\mathbf{T} \cdot {}^{M_2}_{W}\mathbf{T} $$

Due to markers are rigidly attached with the camera, we have ${}^{M_1}_{C_1}\mathbf{T} = {}^{M_2}_{C_2}\mathbf{T} = {}^{M}_{C}\mathbf{T}$, and the above formula can be simplified to:

$$ {}^{C_2}_{C_1}\mathbf{T} \cdot {}^{C}_{M}\mathbf{T} = {}^{C}_{M}\mathbf{T} \cdot {}^{M_2}_{M_1}\mathbf{T} \tag{10} $$

where ${}^{C_2}_{C_1}\mathbf{T} = {}^{C_2}_{B}\mathbf{T} \cdot {}^{C_1}_{B}\mathbf{T}^{-1}$ and their values can be obtained by the method mentioned in Section II-A, ${}^{M_2}_{M_1}\mathbf{T} = {}^{M_2}_{W}\mathbf{T} \cdot {}^{M_1}_{W}\mathbf{T}^{-1}$ and their values are provided by the motion capture system.

A. Hand-eye calibration

Obviously, the equation (10) is the famous hand-eye calibration problem, it comes down to solve the $\mathbf{AX} = \mathbf{XB}$ equation where $\mathbf{A}, \mathbf{B}, \mathbf{X}$ are 4x4 homogeneous transformation matrix. And the equation (10) can be farther decomposed into two equations: a rotation equation and a vector equation depending both on rotation $\mathbf{R}$ and translation $\mathbf{t}$:

$$ \mathbf{R}_A \mathbf{R}_X = \mathbf{R}_X \mathbf{R}_B \tag{11} $$

$$ (\mathbf{R}_A - \mathbf{I})\mathbf{t}_X = \mathbf{R}_X \mathbf{t}_B - \mathbf{t}_A \tag{12} $$

where $\mathbf{I}$ is the 3x3 identity matrix. In this paper, the method based on quaternions, presented in [25], is used for the hand-eye calibration because of its robustness with noise data. In short, the authors use mathematical techniques to transform the hand-eye problems into convex optimization problems that can be easily solved by using related project like CVXPY [26].

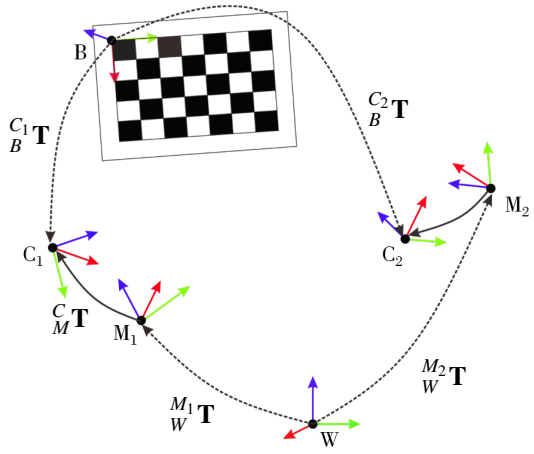


Fig. 4: Intrinsic and extrinsic calibration scene. Camera frame move from C_1 to C_2, and the markers which attached with the camera rigidly move from M_1 to M_2 correspondingly. W is the world frame of MoCap system and frame B is fixed on the chessboard pattern. During the calibration, we keep the chessboard motionless.

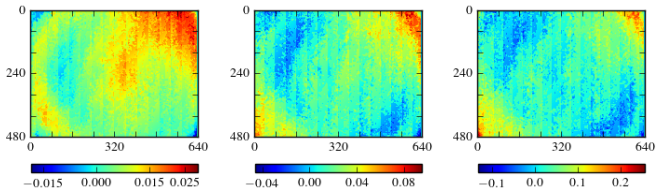
B. Global refinement

The hand-eye calibration accuracy is limited mostly due to the fact that the assumed pose-trajectories are estimated individually and keep fix when aligning them to find the hand-eye transformation. So in this subsection, we proposed a joint maximum likelihood optimization of calibration and trajectory given the measurements allows higher accuracy. This optimization is hard to be solved as a global problem but using the results from Section III-A as an initial guess, a local likelihood maximization can improve the accuracy of the calibration. We use Lie group valued B-splines [27] to represent continuous-time SO(3)-trajectory and use an extension of Levenberg-Marquardt to Lie groups to solve the nonlinear optimization, as in [28].

But in our calibration scene, the values of transformation ${}^{M}_{W}\mathbf{T}$ measured by the motion capture system are very accurate and the main source of the calibration error is the estimated values of transformation ${}^{C}_{B}\mathbf{T}$. So in this work, we model the trajectory for the moving marker frame, M, ${}^{M}_{W}\mathbf{T}(t) =: \mathbf{X}(t)$. And the negative log likelihood function l mentioned in [28] is modified as following:

$$ \begin{aligned} l &= l_1(\mathbf{X} | ({}^{M_i}_{W}\mathbf{T}, t_{M_i})_{i=1}^{l}) + l_2({}^{C}_{B}\mathbf{T}, {}^{C}_{M}\mathbf{T} | ({}^{C_i}_{B}\mathbf{T}, t_{C_i})_{i=1}^{k}) \\ &= \sum_{i=2}^{k_M} \rho(||d_M({}^{M_{i-1}}_{W}\mathbf{T}, {}^{M_i}_{W}\mathbf{T}, {}^{M}_{W}\mathbf{T}(t_{M_{i-1}}), {}^{M}_{W}\mathbf{T}(t_{M_i}))||^2_{\Sigma_M}) \\ &+ \sum_{i=2}^{k_C} \rho(||d_C({}^{C_{i-1}}_{B}\mathbf{T}, {}^{C_i}_{B}\mathbf{T}, {}^{C}_{B}\mathbf{T}(t_{C_{i-1}}), {}^{C}_{B}\mathbf{T}(t_{C_i}))||^2_{\Sigma_C}) \end{aligned} \tag{13} $$

where ${}^{C}_{B}\mathbf{T}(t) := {}^{W}_{B}\mathbf{T} \cdot {}^{M}_{W}\mathbf{T}(t) \cdot {}^{C}_{M}\mathbf{T}$, ${}^{M}_{W}\mathbf{T}(t) = \mathbf{X}(t)$, $\rho(s) = log(1+s)$, d_M and d_C are relative, $d(\mathbf{A}, \mathbf{A}', \mathbf{B}, \mathbf{B}') = d(\mathbf{A}^{-1}\mathbf{A}', \mathbf{B}^{-1}\mathbf{B}')$. As displacement vector $d(\mathbf{A}, \mathbf{B}) \in \mathbb{R}^6$ on SE(3), we use coordinates of $(log_{SO(3)}(\mathbf{R}), \mathbf{u})$ with respect to a fixed positive orthonormal basis, where $\mathbf{u}$ is a translation and $\mathbf{R}$ is a proper rotation such that (uniquely) $\mathbf{u} \circ \mathbf{R} := \mathbf{A}^{-1} \cdot \mathbf{B}$.We first optimize



(a) 1±0.01m (b) 2±0.01m (c) 3±0.01m

Fig. 5: Depth error maps. Note that the color scale is different for each map.

l_1 to estimate the trajectory function $\mathbf{X}(t)$ of marker frame. And then optimize l_2 to calculate the optimal estimated value of ${}^{C}_{M}\mathbf{T}$.

IV. Depth Correction

Many depth sensors have *myopic* property which means that an incorrect parameter set results in an error that increases with distance. To make matters worse, the errors of depth measurements at different pixel positions are also different (as shown in Fig. 5). To model the effects of this distortion, in the latest work [13], the authors separate the error into two different parts and treat them separately, including *distortion*, the error responsible of the local alteration of the object shape, and *global error*, the systematic wrong estimation of the average depth. That is, the real depth d^* is estimated as

$$d^* = (g \odot u)_{(u,v)}(d) \tag{14}$$

where function $u(\cdot)$ takes into account the local distortion and function $g(\cdot)$ makes a global correction of the depth measures.

In this work, the method mentioned in Section III is used to get the poses of the IR camera in real time. in this section, firstly, we will show how to track a plane to get the ground truth of the depth measurements. And then we propose a sparse Gaussian process method to model the posterior probability $P(d^*|d)$, which means that both the mean $\mu(d^*)$ and variance $\sigma^2(d^*)$ of d^* at different d values are obtained.

A. Plane frame tracking

First of all, we want to track a plane frame P, it means that the transformation ${}^{P}_{W}\mathbf{T}$ can be obtained in real time. We attach three reflective markers A, B and C on a board or just use the official calibration pattern which shown in Fig. 2(c), the plane which contains these three markers is parallel to the board. And MoCap system provides us the three-dimensional coordinates of the three markers in real time. $P_A = [x_A, y_A, z_A]^T$, $P_B = [x_B, y_B, z_B]^T$ and $P_C = [x_C, y_C, z_C]^T$. The following shows how to calculate the transformation matrix from world frame W to the frame P which formed by these markers.

The first step and without loss of generality, we choose the unit direction vector $\vec{x}$ of $\overrightarrow{AB}$ as the direction of the X axis of frame P. We can also find the plane unit normal vector $\vec{z}$ (as well as the direction of the Z axis) by using the geometric properties $\vec{z} \perp \overrightarrow{AB}$ and $\vec{z} \perp \overrightarrow{AC}$. And the direction of the Y axis can be estimated as $\vec{y} = \vec{z} \times \vec{x}$. If there is no measurement error of MoCap system and placement deviation of markers, the 3x3 rotation matrix $\mathbf{R}$ from frame W to the plane frame P can be easily obtained :

$$\mathbf{R} = (\vec{x}, \vec{y}, \vec{z}) \tag{15}$$

However, the actual three direction vectors are not perpendicular to each other. Therefore the singular value decomposition (SVD) $\mathbf{R} = \mathbf{U}\Sigma\mathbf{V}^T$ is used to get the approximate estimate $\mathbf{R}'$ of the rotation matrix $\mathbf{R}$ (the detailed proof can be found in [21]). That is

$$\mathbf{R}' = \mathbf{U} \cdot \mathbf{V}^T \tag{16}$$

Finally, the 4x4 transformation matrix from the frame W to frame P is:

$${}^{P}_{W}\mathbf{T} = \begin{pmatrix} \mathbf{R}' & P_A \\ \mathbf{0} & 1 \end{pmatrix} \tag{17}$$

And in order to filter out the depth data that is projected outside the plane, we also need to provide the plane range $(x_{min}, x_{max}, y_{min}, y_{max})$, which does not need to be very accurate as long as it is guaranteed to be within the actual plane range.

B. Data acquisition

The plane mentioned in the previous section can be chosen as a large plane that can not be moved (like a wall), or a smaller plane that can be freely moved (as shown in Fig. 2(e)). This is because we can track the 3D pose of the plane and the camera in real time using the method mentioned in Section III and IV-A. However, it is obvious that larger the plane, more efficient data acquisition.

We firstly record the depth images of the Kinect at different distances with the plane as well as ${}^{P}_{W}\mathbf{T}$ and ${}^{M}_{W}\mathbf{T}$ at the same time, and then process the recorded data offline to generate datasets for training GP models. This off-line process is briefly described in algorithm 1. Firstly, we use ${}^{P}_{I}\mathbf{T}$ to restore the plane equation $\mathbf{P}$ ($l.$ 6, where $plane()$ is a function which get the plane equation of the XY axis). And for a given pixel coordinate $\mathbf{m} = (u, v)$ in a depth map, we can get the corresponding pixel coordinates in the IR image by using the formula (9) ($ll.$ 8-9). And then the corresponding 3D point $\mathbf{M} = (x, y, 1)$ is calculated by using formula (5)-(7) ($l.$ 10), which means a beam of radiation from the optical center to $\mathbf{M}$. We can find the intersection $\mathbf{M}'$ of the ray and the plane $\mathbf{P}$ ($l.$ 11). Finally, we use the plane range $(x_{min}, x_{max}, y_{min}, y_{max})$ to determine whether the intersection point falls in the plane and decide whether to keep it in the dataset Ψ or not ($ll.$ 12-15) (as illustrated in Fig. 6).

C. Heteroscedastic Gaussian Processes

Gaussian Processes are a generalization of normal distributions to functions, describing functions of finite-dimensional random variables. Compared with the curving fitting method mentioned in [13] , the Gaussian processes does not need to know the specific expression of model in advance.

The standard GP formulation assumes that each data pair $(\mathbf{x}_i, \mathbf{y}_i)$ is drawn from a process with i.i.d Gaussian noise:

$$\mathbf{y}_i = f(\mathbf{x}_i) + \varepsilon \tag{18}$$

where ε is the noise generated from a Gaussian distribution with known static variance σ_n^2.

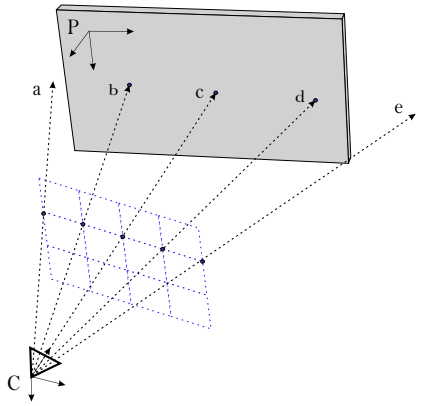


Fig. 6: Brief sketch of the projection procedure of each pixel point in IR image onto the tracking plane P using the model of IR camera. And 3D point of intersection a and e are out of bounds and should be filtered out.

Algorithm 1: Data acquisition

Input: ${}^{P}_{W}\mathbf{T}$, ${}^{M}_{W}\mathbf{T}$, measured depth images $\mathbf{D}$;
Output: Dataset Ψ for training GP models;

```
I_M T = hand_eye_calibration();
Initialize plane range (x_min, x_max, y_min, y_max);
Initialize IR image to depth image offset u', v';
for i ∈ range(D) do
    P_I T_i = P_W T_i · M_W T_i^{-1} · I_M T^{-1};
    P_i = plane(P_I T_i);
    for pixel(u, v) ∈ D_i.size() do
        u_I = u + u';
        v_I = v + v';
        M_I = project(u_I, v_I);
        M'_I = cross(M_I, P_i);
        M'_P = P_I T · M'_I;
        if (M'_P.x ∈ [x_min, x_max]) ∧ (M'_P.y ∈ [y_min, y_max]) then
            Ψ = Ψ ∪ {(u, v, D_i(u, v), M'_I.z)}
        end
    end
end
return Ψ;
```

However, this formulation ignores the characteristics of varying variance $\sigma^2(d^*)$ in this work. In order to incorporate this information into the system, Goldberg *et al.* [29] changed the first assumption by considering that each data pair $(\mathbf{x}_i, \mathbf{y}_i)$ is drawn from a process with known variance that depends on $\mathbf{x}_i$. That is:

$$\mathbf{y}_i = f(\mathbf{x}_i) + \varepsilon(\mathbf{x}_i) \tag{19}$$

where $\varepsilon(\mathbf{x}_i)$ is the noise generated from a Gaussian distribution with input-dependent variance $\sigma_n^2(\mathbf{x}_i)$. In this work, the open source project GPz [20] is used to train a set of sparse Gaussian Processes for heteroscedastic posterior probability $P(d^*|d)$. To illustrate the differences between standard GP formulation and the heteroscedastic one, we show the predictions generated using a toy function $y = sin(x)$ with variance $v = (x+1)/15$. Fig. 7 shows a comparison of the predictions using a standard GP and the proposed approach which successfully models the toy-function variance.

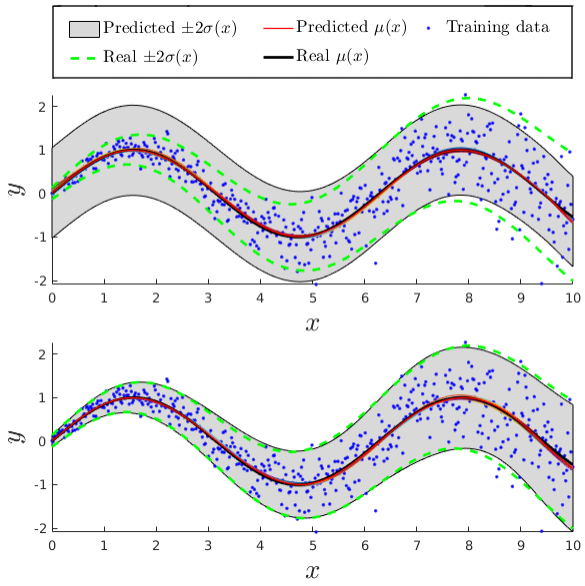


Fig. 7: Prediction generated for a toy function $y = sin(x)$ with variance $v = (x+1)/15$, top is the standard GP and bottom is the heteroscedastic GP.

V. Implementation

The General Batch-Calibration Framework, developed by the authors in [15], employs the optical motion capture system (as shown in Fig. 2(a).) as an automatic measuring system. And our MoCap system consists of 12 cameras equipped with infrared LED around the camera lens and it can track frame-to-frame 3D positions of the reflective markers in real time. The measurements of MoCap and robots are sent through the MoCap Bridge module and ROS[1] Bridge module and the NTP module is responsible to synchronize time between them. Some large quantities of data could be stored in local and processed offline.

We calibrate the motion capture system using the Motive software provided by Optitrack [30]. The calibration procedure requires waving a calibration stick (as illustrated in Fig. 2(b).) with three markers extensively through the motion capture area. The Motive software computes the poses of the motion capture cameras from these point correspondences. After that, we can place a right-angle calibration tool (as shown in Fig. 2(c).) to settle down the world coordinate system W. To validate the results of this calibration procedure, we hold a stick which contains two markers with a distance of 1 m approximately and move freely in the MoCap area, we measured a standard deviation of 0.99 mm. And then we left the markers at the reference positions and covered the markers

[1] http://www.ros.org/

after recording them so that the MoCap system lost track and the markers have not been moved in reality. After a while, we removed the covers and estimated the pose of the markers again, we always found a position error of slight below 0.01 mm. we think that the position estimates of our MoCap system are highly accurate and stable over time.

VI. Experiments

In this section, we want to show that our method is able to provide more accurate calibration results for Kinect-like sensors. We independently analyze the results of our hand-eye calibration (Section III) and depth correction (Section IV). Then we report some real experiments of an RGB-D visual SLAM applied to a mobile robot which mounted a Kinect device, where we show that the accuracy of positioning and mapping highly benefits from using RGB-D data calibrated with our calibration approach. Finally, performance comparison with other related work is shown at the end.

A. Hand-eye calibration

The accuracy of hand-eye calibration results are difficult to evaluate, so we evaluate the whole effect of the intrinsic and extrinsic calibration in this work by means of the reprojection error. We placed six markers as accurately as possible on the outer corners of the checkerboard (as shown in Fig. 2(e)), such that the transformation between the visual checkerboard and the motion capture markers is known. Given these points observations and the point model, we can compute its pose ${}^{W}_{B}\mathbf{T}'$ with respect to the world frame of the motion capture system by means of the iterative closest point method (ICP). For every 3D position M_{ij} of corner j in image i, we can calculate its corresponding coordinates in the camera coordinate system and compare the deviation from the pixel coordinates m_{ij} of detected corner. That is

$$e_{ij} = repr({}^{C}_{M}\mathbf{T} \cdot {}^{M}_{W}\mathbf{T}_i \cdot {}^{W}_{B}\mathbf{T}'_i \cdot M_{ij}) - m_{ij}$$

We recorded 44 8x5 checkerboard images which have a total of 1760 corners. Fig. 8(a) plots the corner reprojection error e_{ij} with the value of ${}^{C}_{M}\mathbf{T}$ estimated in Section III-A and the results by using the global refinement method in Section III-B are illustrated in Fig. 8(b). Fig. 8(c) shows the distribution of reprojection errors $\left|e_{ij}\right|$. All of the figures in Fig. 8 prove that the refinement step mentioned in Section III-B give a more accurate estimation of extrinsic parameters. Another thing needs to mention is that the reprojection errors here are not only caused by inaccurate estimation of ${}^{C}_{M}\mathbf{T}$, but also by the inaccurate position of attached markers, inaccurate corner detection and the measurement error of MoCap system. The reprojection errors of our method are within 2 pixels, and we think that the accuracy is acceptable for depth correction. Some qualitative analyses are also provided in Fig. 9, which show that more accurate fusion of color and depth informations result from better alignment between RGB and depth cameras with our estimated value of ${}^{R}_{I}\mathbf{T}$.

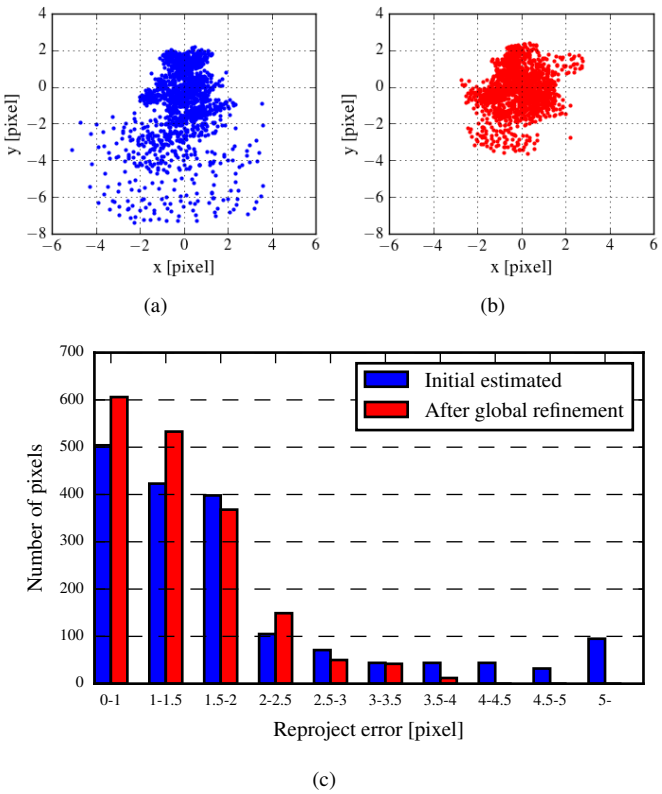


Fig. 8: Reprojection error of all pixels in all recorded images: (a) is the errors without global refinement and (b) is the final result with our global refinement, (c) the distribution of reprojection errors.

B. Depth correction

For validating the performance of our proposed depth correction method, we collected two datasets (include depth and color image pair, ${}^{M}_{W}\mathbf{T}_i$ and ${}^{P}_{W}\mathbf{T}_i$): a training set (contains 597 image pairs and used to generate the correction functions) and a testing set (contains 428 image pairs and used to evaluate the calibration accuracy). After deploying our depth correction algorithm on the training set, we obtained 480x640 depth-correct GP models. And in order to compare with the state-of-the-art depth correction method [13], supposing the depth error is corrected by a second degree polynomial, we also obtained 480x640 functions which contain 480x640x3 parameters totally (some examples are illustrated in Fig. 10). And then we use these models or functions to correct our depth images in testing dataset. The total depth errors are plotted in Fig. 11 where plot 11(a) is original depth errors, plot 11(b) is the depth errors with curve fitting method [13] and plot 11(d) is the corrected depth errors by using the mean value provied by our GP models. And the standard deviation of depth error learned by our GP method is also plotted in Fig. 11(c). We can obviously find that our method remarkably reduce the error of Kinect depth measurements and is better than the curve fitting method. And in Fig. 9, we choose three single point cloud which is keep a distance of 1m, 2m and 3m with a planar surface and we can find that the three point cloud become more even after corrected by our method compared

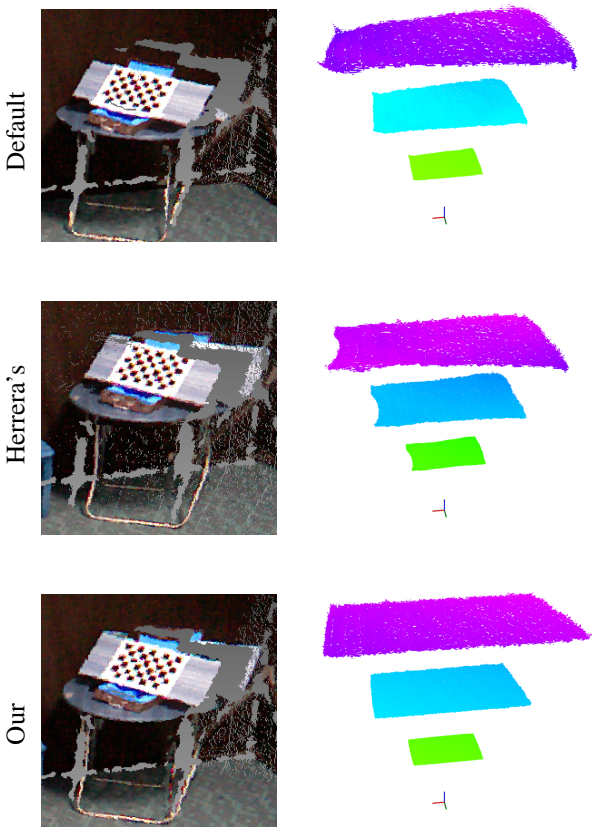


Fig. 9: some results of our calibration approach: first column: the coloured point cloud resulted from different alignments between the RGB camera and the depth camera. second column: three point clouds (approximately with distance of 1m, 2m, 3m) of a planar surface without or with different depth correction.

with Herrera's method [7], and Fig. 12 is about their top and front views and further shows the effect of our proposed depth correction algorithm.

C. Visual SLAM

As further validation of our method, we report an experiment performed using our home service robot (Kejia F2) with an RGB-D visual SLAM system (RGB-D SLAM v2 [1]). And we show how the accuracy of positioning using RGB-D data calibrated with our method by means of evaluation metrics: absolute trajectory error (ATE) and relative pose error (RPE) [14], and the final 3D point cloud maps have also been presented.

We moved our mobile robot forward by 2m at 0.02 m/s, then turn 90 degrees and moved forward 3m. And at the same time, we record the poses of markers attached to the Kinect and we can obtain the trajectories of the Kinect using the result of hand-eye calibration ${}^{C}_{M}\mathbf{T}$, and we think of it as ground truth of Kinect trajectory. Table I shows the root mean square error (RMSE) of ATE and RPE for the experiment: the reduce of ATE is remarkable and the RPE also have

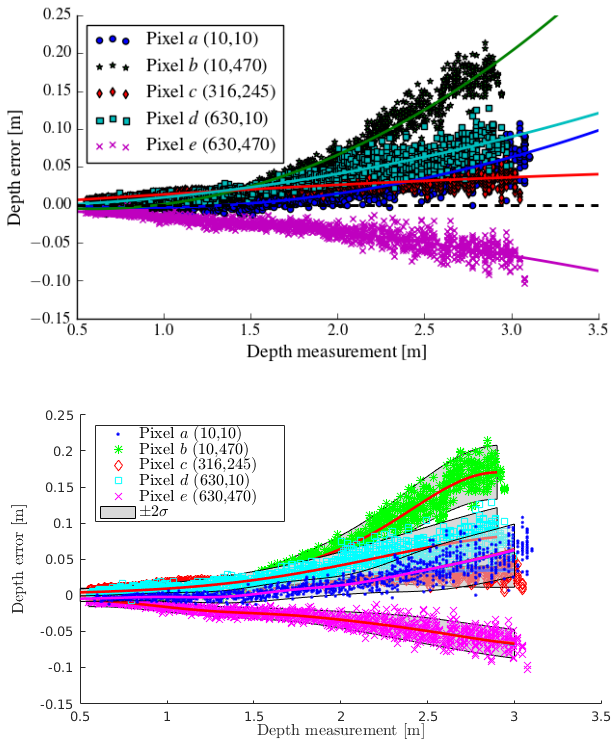


Fig. 10: Depth error and corresponding polynomial fitting curve (top) or GP model (bottom) of five different pixel coordinates (pixel c is the coordinate of optical center). Note that different pixel coordinates have different parameters of fitting curves or GP models.

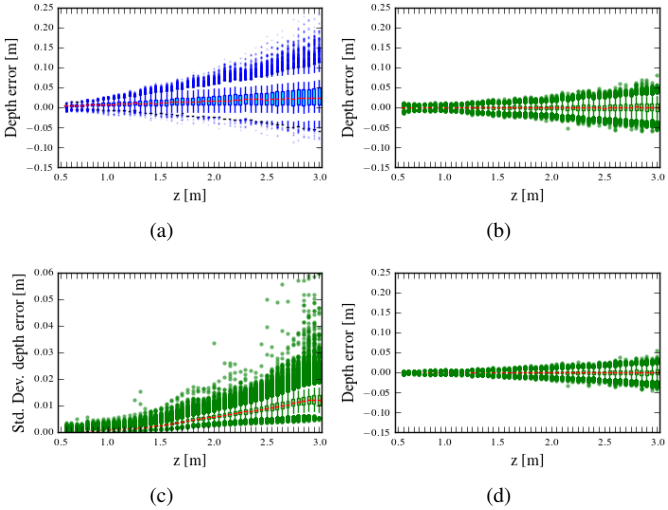
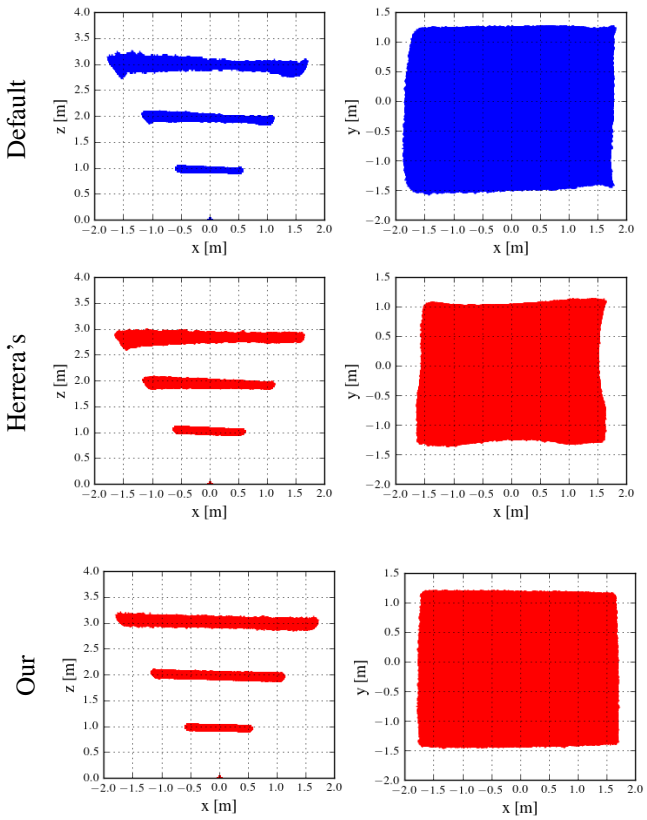


Fig. 11: Depth error in different measure distance. (a) is the original data, (b) is the corrected data with second degree polynomial depth correction method, (c) is the standard deviation of depth error learned by our GP method and (d) is the corrected data by using the mean value of depth error provided by our GP models



(a) Top view of the point clouds (b) front view of the point cloud

Fig. 12: Different views of point cloud. First column: side views of three point clouds which approximately keep distance of 1m, 2m and 3m from a planar surface. Second column: front views of the 3m point clouds, and we can easily find that the bottom one (corrected with our method) is closer to the rectangle than others.

slightly reduced. Fig. 13(a) shows a top view of the estimated and ground truth trajectories without or with our proposed calibration approach, along with the generated point cloud maps (Fig. 13(b)). The trajectory estimated using the corrected data is obviously closer to the ground truth compared to the one estimated using the original data, and the quality and the precision of the reconstructed point cloud map is also improved by using the corrected data.

TABLE I: RMSE of ATE and RPE

	Original data	Corrected data
ATE [m]	0.092	**0.041**
RPE.translation [m]	0.0082	**0.0073**
RPE.rotation [deg]	0.3576	**0.3517**

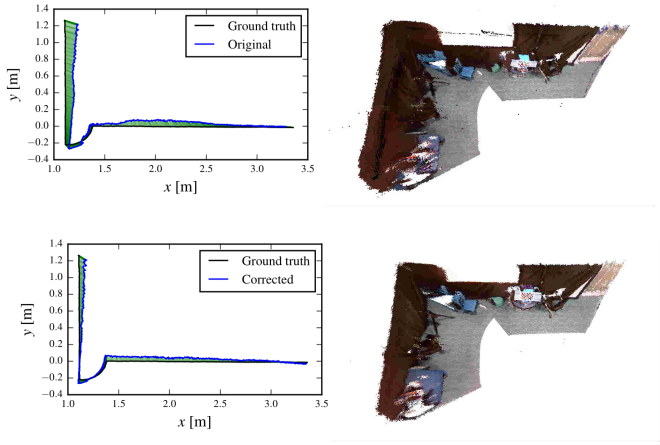
D. Performance comparison

In this subsection, we compare the calibration accuracy of our system against two state-of-the-art calibration methods, the one from Basso *et al.* [13] and the one from Herrera *et al.* [7], using the original implementations provided by the authors. For each method, we acquired large training sets and adopted each calibration approach. And for comparing the calibration accuracy, we collected a test set framing a big reference hollow cube (Fig. 14(a)) with large AprilTags [31] attached to each visible side. Using this 3D pattern, it is possible to accurately compute the plane equations of the three sides using the obtained AprilTag images. we use these planes, their intersection point x as ground truth data. For each tested method, we estimated the plane equations by fitting the three planes to the corrected point clouds, computing also their intersection point x'. To further prove the role of the probability distribution in the state estimation problem, we record three depth maps in each of the same locations. The probability distribution of the true depth d^* is:

$$P(d^*|d_1,d_2) = P(d^*|d_1)\cdot P(d^*|d_2)\cdot P(d^*|d_3)$$

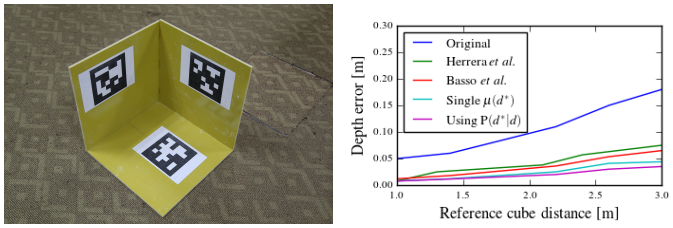
where d_1, d_2 and d_3 are the depth measurements of intersection point x in the three depth maps. Since all probability distributions are approximating Gaussian distribution, we have:

$$P(d^*|d_i)\cdot P(d^*|d_j) = N\left(\frac{\sigma^2(d_i)\cdot\mu(d_j)+\sigma^2(d_j)\cdot\mu(d_i)}{\sigma^2(d_i)+\sigma^2(d_j)}, \frac{\sigma^2(d_i)\cdot\sigma^2(d_j)}{\sigma^2(d_i)+\sigma^2(d_j)}\right)$$



(a) Top view of the trajectories (b) Point cloud map

Fig. 13: Qualitative results of the RGB-D SLAM v2 experiments: (first row) The result of using the original data; (second row) The result of using the corrected data.



(a) (b)

Fig. 14: (a) The reference hollow cube used as a ground truth in the performance evaluations; (b) Accuracy comparison in the reference cube localization for increasing depths.

where $\mu(d)$ and $\sigma^2(d)$ are the mean and variance values, which can be obtained from our GP models. Fig. 14(b) shows the depth errors of different methods at different distances of the reference cube.

VII. Conclusion

In this paper, we present a reliable and accurate method to calibrate the Kinect-like sensors using motion capture system. The proposed calibration procedure can not only provide the intrinsic and extrinsic parameters of each camera, but also generate a set of pixel-wise depth correction models based on heteroscedastic Gaussian Processes. In the intrinsic calibration, we completely calibrated the pixel offset between the depth image and IR image. In our extrinsic calibration, we make full use of the characteristics of the MoCap system to customize the global refinement step for the hand eye calibration, and it further improves the accuracy of the extrinsic calibration. Moreover, a non-recursive and novel data acquirement method is used to get the ground truth of every pixel in depth images and a one-step, model-free depth correction approach is applied to obtain the parameters of depth correction model set. The proposed calibration method achieves better results than previous related works and the entire calibration results have greatly improved positioning and mapping of visual SLAM tasks using the Kinect sensors.

References

[1] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[2] B. Tian, V. A. Shim, M. Yuan, C. Srinivasan, H. Tang, and H. Li, "Rgb-d based cognitive map building and navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1562–1567.

[3] L. Bo, X. Ren, and D. Fox, "Learning hierarchical sparse features for rgb-(d) object recognition," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 581–599, 2014.

[4] G. Alenyà, S. Foix, and C. Torras, "Using tof and rgbd cameras for 3d robot perception and manipulation in human environments," *Intelligent Service Robotics*, vol. 7, no. 4, pp. 211–220, 2014.

[5] Technical description of kinect calibration. [Online]. Available: http://wiki.ros.org/%20kinect_calibration/technical/

[6] Ros freenect launch for kinect. [Online]. Available: http://wiki.ros.org/freenect_launch

[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*, vol. 34, no. 10, pp. 2058–2064, 2012.

[8] J. Jung, Y. Jeong, J. Park, H. Ha, J. D. Kim, and I.-S. Kweon, "A novel 2.5 d pattern for extrinsic calibration of tof and camera fusion system," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011, pp. 3290–3296.

[9] A. N. Staranowicz, G. R. Brown, F. Morbidi, and G.-L. Mariottini, "Practical and accurate calibration of rgb-d cameras using spheres," *Computer Vision and Image Understanding*, vol. 137, pp. 102–114, 2015.

[10] J. Smisek, M. Jancosek, and T. Pajdla, "3d with kinect," in *Consumer depth cameras for computer vision*. Springer, 2013, pp. 3–25.

[11] C. Zhang and Z. Zhang, "Calibration between depth and color sensors for commodity depth cameras," in *Computer vision and machine learning with RGB-D sensors*. Springer, 2014, pp. 47–64.

[12] A. Teichman, S. Miller, and S. Thrun, "Unsupervised intrinsic calibration of depth sensors via slam." in *Proceedings of the Robotics: Science and Systems*, vol. 248, 2013, p. 3.

[13] F. Basso, A. Pretto, and E. Menegatti, "Unsupervised intrinsic and extrinsic calibration of a camera-depth sensor couple," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6244–6249.

[14] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[15] K. Zheng, Y. Chen, F. Wu, and X. Chen, "A general batch-calibration framework of service robots," in *Proceedings of the International Conference on Intelligent Robotics and Applications*. Springer, 2017, pp. 275–286.

[16] A. Perez-Yus, E. Fernandez-Moral, G. Lopez-Nicolas, J. Guerrero, and P. Rives, "Extrinsic calibration of multiple rgb-d cameras from line observations," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 273–280, 2018.

[17] Y. M. Kim, D. Chan, C. Theobalt, and S. Thrun, "Design and calibration of a multi-view tof sensor fusion system," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2008, pp. 1–7.

[18] C. Mei and P. Rives, "Calibration between a central catadioptric camera and a laser range finder for robotic applications," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 532–537.

[19] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 4164–4169.

[20] I. A. Almosallam, M. J. Jarvis, and S. J. Roberts, "Gpz: non-stationary sparse gaussian processes for heteroscedastic uncertainty estimation in photometric redshifts," *Monthly Notices of the Royal Astronomical Society*, vol. 462, no. 1, pp. 726–739, 2016.

[21] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[22] W. Xiang, C. Conly, C. D. McMurrough, and V. Athitsos, "A review and quantitative comparison of methods for kinect calibration," in *Proceedings of the 2nd international Workshop on Sensor-based Activity Recognition and Interaction*. ACM, 2015, p. 3.

[23] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[24] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of hough transform methods for circle finding," *Image and vision computing*, vol. 8, no. 1, pp. 71–77, 1990.

[25] Z. Zhao, "Hand-eye calibration using convex optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2947–2952.

[26] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[27] H. Sommer, J. R. Forbes, R. Siegwart, and P. Furgale, "Continuous-time estimation of attitude using b-splines on lie groups," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 242–261, 2015.

[28] F. Furrer, M. Fehr, T. Novkovic, H. Sommer, I. Gilitschenski, and R. Siegwart, "Evaluation of combined time-offset estimation and hand-eye calibration on robotic datasets," in *Proceedings of the Field and Service Robotics*. Springer, 2018, pp. 145–159.

[29] P. W. Goldberg, C. K. Williams, and C. M. Bishop, "Regression with input-dependent noise: A gaussian process treatment," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 1998, pp. 493–499.

[30] Calibration of optitrack mocap. [Online]. Available: https://v20.wiki.optitrack.com/index.php?title=Calibration

[31] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4193–4198.