Autonomous Robots (2020) 44:1485–1503
https://doi.org/10.1007/s10514-020-09941-w

RGB-D camera calibration and trajectory estimation for indoor mapping

Liang Yang[2,3] · Ivan Dryanovski[1] · Roberto G. Valenti[2] · George Wolberg[2] · Jizhong Xiao[2]

Received: 3 April 2019 / Accepted: 27 July 2020 / Published online: 17 August 2020

Abstract
In this paper, we present a system for estimating the trajectory of a moving RGB-D camera with applications to building maps of large indoor environments. Unlike the current most researches, we propose a 'feature model' based RGB-D visual odometry system for a computationally-constrained mobile platform, where the 'feature model' is persistent and dynamically updated from new observations using a Kalman filter. In this paper, we firstly propose a mixture of Gaussians model for the depth random noise estimation, which is used to describe the spatial uncertainty of the feature point cloud. Besides, we also introduce a general depth calibration method to remove systematic errors in the depth readings of the RGB-D camera. We provide comprehensive theoretical and experimental analysis to demonstrate that our model based iterative-closest-point (ICP) algorithm can achieve much higher localization accuracy compared to the conventional ICP. The visual odometry runs at frequencies of 30 Hz or higher, on VGA images, in a single thread on a desktop CPU with no GPU acceleration required. Finally, we examine the problem of place recognition from RGB-D images, in order to form a pose-graph SLAM approach to refining the trajectory and closing loops. We evaluate the effectiveness of the system on using publicly available datasets with ground-truth data. The entire system is available for free and open-source online.

Keywords RGB-D · Computer vision · 3D mapping · Camera calibration

1 Introduction

An RGB-D camera is a device which provides two concurrent image streams: a conventional color image, and a depth image, containing a measure of the distance from the camera to each observed pixel along the optical axis. The two images can be used together to obtain a dense, textured 3D model of the observed scene.

The properties of RGB-D data, together with the low cost of current devices, have made RGB-D cameras very popular among the computer vision and robotics communities. RGB-D data has been used in various applications, including visual odometry, SLAM, scene modeling, and object recognition.

Liang Yang and Ivan Dryanovski have contribute equally to this paper.

This work is supported in part by U.S. Army Research Office under Grant No. W911NF-09-1-0565, U.S. National Science Foundation under Grants No. IIS-0644127 and No. CBET-1160046, Federal High-Way Administration (FHWA) under Grant Nos. DTFH61-12-H-00002 and PSC-CUNY under Grant No. 65789-00-43.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10514-020-09941-w) contains supplementary material, which is available to authorized users.

✉ Jizhong Xiao
jxiao@ccny.cuny.edu

Liang Yang
lyang1@ccny.cuny.edu

Ivan Dryanovski
idryanovski@google.com

Roberto G. Valenti
robertogl.valenti@gmail.com

George Wolberg
wolberg@cs.ccny.cuny.edu

1 Department of Computer Science, The Graduate Center, The City University of New York, 365 Fifth Avenue, New York, NY 10016, USA

2 The City College of New York, Convent Ave & 140th Street, New York, NY 10031, USA

3 Shenyang Institute of Automation, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Shenyang, China

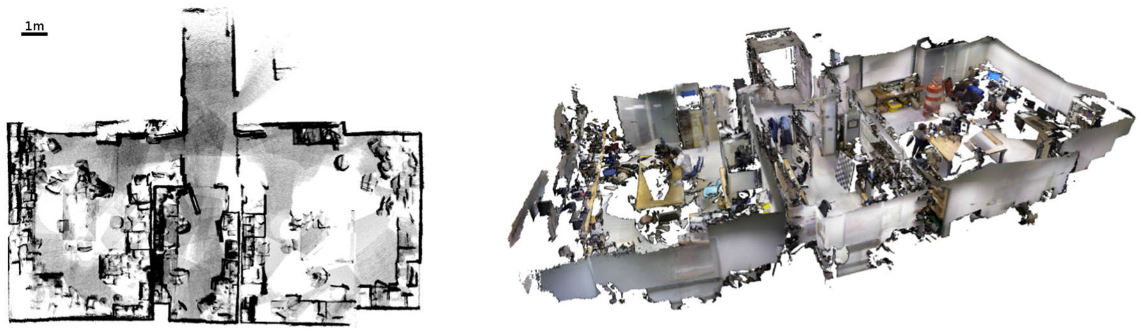


Fig. 1 Top-down and side views of 3D map constructed from RGB-D data using our system

In this paper, we present our work on 3D localization and mapping of indoor environments with an RGB-D sensor, running on computationally constrained platforms. An example of a 3D reconstruction produced by our system is shown in Fig. 1. Our research has been motivated from the perspective of mobile robotic perception; therefore, there is a strong focus throughout our work on computational efficiency and real-time performance. The three key problems we address are the depth calibration and noise estimation of the point cloud, the pose estimation of a moving camera using 'feature model', and re-localization from RGB-D images for detecting and dealing with trajectories revisiting the same location. We believe our contribution to the first two areas are applicable beyond the field of robotic perception, to anyone working on indoor modeling with RGB-D data. Our place recognition work is primarily included for the sake of system completeness and as a straightforward method for offline post-processing; we don't claim any significant contribution over more established methods in the field.

We begin by examining the *accuracy* and *precision* of depth images produced by RGB-D cameras. Accuracy refers to the closeness of the measurements to the truth, while precision refers to the variation of repeated measurements under the same conditions. Compared to their more expensive range-sensing counterparts such as laser scanners, RGB-D cameras have lower precision and accuracy. Consider Fig. 2, which shows the result of a simple experiment where we placed an RGB-D camera at three different distances away from a flat wall. For the top figure, the point cloud visible in the figure is the result of aggregating multiple depth readings at 4 m away. An ideal sensor would produce a thin, straight line as the top view of the point cloud. The bottom right figure shows the result taken at 8.0 m away, which is even worse with an average 578.9 mm error. However, we notice that the left bottom figure shows the result taken at 1.5 m away, which only produce an average 8.5 mm error. It is immediately obvious that the real camera has not only a high uncertainty, but also a high degree of systematic error which is highly associated with the distance to the target. More importantly, the

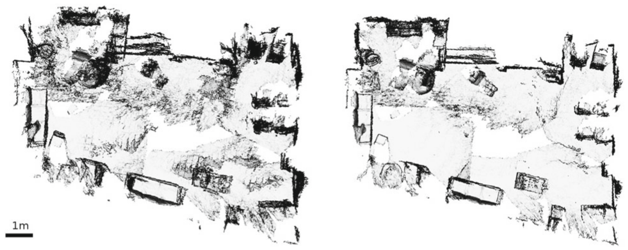


Fig. 2 Top-down view of a point cloud obtained by measuring a flat wall using an RGB-D camera. Top image: the point cloud is constructed by averaging multiple depth images to remove the effect of random noise. The curvature in the point cloud indicates that the depth readings (Z) for each pixel have a strong systematic error. Bottom left: the raw point cloud constructed at 1.5 m, the average depth error is 8.5 mm. Bottom right: the raw point cloud constructed at 8.00 m, the average depth error is 578.9 mm

systematic error varies across different sections of the image, producing geometric distortions.

In order to solve the above problem, we propose a per-pixel polynomial model calibration technique to remove the bias in the depth readings. The calibration technique relies on observing a flat checkerboard at the initial position and then estimate the transformation using ORB-SLAM2 (Mur-Artal and Tardós 2017) algorithm (which is proved to have smaller than 1.0% accuracy). A *ground truth* pose is determined by calculating the transformation from the camera to the chessboard coordinate system. A plane fitting algorithm is deployed to find the wall based on the first RGB-D frame, then we only need to compare the accuracy between the measurement and the fitted plane. With this model, incoming depth images can be unwarped so that the bias in each pixel is removed. We are also inspired by Basso et al. (2018) to improve the implementation of the depth calibration for real time application purpose, but we do not compromise with lowering the depth resolution. We demonstrate experimentally that unwarping the data has a significant impact on the performance on visual odometry and mapping algorithms that rely on depth readings. We further formulate an uncertainty model for the depth reading in each pixel. The uncertainty is based on a Gaussian mixture model of depth readings of the pixels in a local neighborhood window. When treating each 3D point as a probabilistic distribution, the calibration technique accurately determines the mean, while the uncertainty model predicts the standard deviation.

The probabilistic measurement model forms the basis of a visual odometry pipeline for trajectory estimation. We begin by computing the locations of sparse features in the incoming RGB-D image, and their corresponding 3D coordinates in the camera frame. Next, we align these features against a global model dataset of 3D features, expressed in the fixed coordinate frame. After calculating the transformation, the model is augmented with the new data. We associate features from the RGB-D image with features in the model, and update them using a Kalman filter framework considering the uncertainty as spatial covariance. Any features from the image which cannot be associated are inserted as new landmarks in the model set.

The model (which starts out empty) gradually grows in size as new data is accumulated. To guarantee constant-time performance, we place an upper bound on the number of points the model can contain. Once the model grows beyond that size, the oldest features are dropped to free up space for new ones. By performing alignment against a persistent model instead of only the last frame, we are able to achieve significant decrease in the drift of the pose estimation. We perform the trajectory estimation in real time, at rates of 30 Hz or higher (the camera outputs the images at 30 Hz, but our algorithm is able to process them faster). It uses a single thread, and does not require a GPU.

The paper is organized as follows. Section 2 reviews the previous work on RGB-D camera calibration and uncertainty analysis. Section 3 describes RGB-D camera measurement model, including the depth image calibration procedure and noise models. Section 4 presents the trajectory estimation pipeline, focusing on the visual odometry algorithm. Section 5 presents our work on place recognition from RGB-D images, and its application to global keyframe-based pose alignment. We present our experimental results using multiple datasets in Sect. 6. Finally, Sect. 7 concludes the paper with a summary of our contributions and areas of potential further work.

2 Previous work

In the field of camera calibration, Smisek et al. (2011) present a calibration procedure for an RGB-D camera, including the intrinsic parameters or the RGB and depth cameras and the extrinsic matrix between them. Further, they examine the relationship between the raw device output and the metric depth, as well as the resolution and quantization of the depth output. They correct for systematic errors in the depth, but the correction is performed using a constant term which is trained only in the 0.7–1.3 m range. Khoshelham and Elberink (2012) discuss the depth accuracy and resolution of an RGB-D camera, including the relationship between the raw depth output and the metric depth. They derive an uncertainty model for the metric depth based on the resolution of the device and the uncertainty of the raw depth. The model considers pixel readings as independent of neighboring pixels.

Park et al. (2012) propose a mathematical uncertainty model for 3D visual features. Further, they propose an alternative model from computing the metric depth from the raw depth output. Olesen et al. (2012) propose an uncertainty model for the depth reading based on a parametric model which considers the radial distance of a pixel from the image center and the depth reading at that pixel. Nguyen et al. (2012) propose a depth noise model by measuring lateral and axial measurement distributions as a function of distance and angle of the sensor to a surface. Herrera et al. (2010) present an algorithm to simultaneously calibrate a depth and RGB camera, including their intrinsic parameters and the pose between them. The method is based on observing checkerboards attached to a large flat surface. Karan (2015) also presented an algorithm to simultaneously calibrate a depth and RGB camera within 4 m range, which using the six order distortion function (Zhang 2000). It does not consider the systematic depth error problem.

In terms of depth image compensation models, our work is most closely related to that of Teichman et al. (2013) and Basso et al. (2018). Teichman et al. (2013) describe a procedure for training an unwarping model based on myopic parameters. Their system uses SLAM-based structure instead of checkerboard images as a source of reference. This has the added advantage of being 4able to use natural scenes for online calibration. However, it couples the model training with an existing SLAM implementation, which in turn may be already affected by reconstruction errors due to the depth image biases. This fact becomes increasingly challenging as the longer the distance, the more severe the uncalibrated biases are. Basso et al. (2018) proposed a similar idea of using pixel-wise correction approach to correct the depth, which also proves that a second-order polynomial could be able to eliminate the depth error. However, it only uses checkerboard based projective-n-point (PNP) method to estimate the pose from the camera to the wall, which heavily affects the calibration result since the pose estimation error grows with the distance to the checkerboard. In particular, we demonstrate results from a depth camera on a mobile device with systematic errors large enough to compromise structure-based calibration methods. To solve this problem, we propose a hybrid pose estimation system combing checkerboard PNP calculation and rapid SLAM to obtain the reference plane detection, which is proved to be more robust and accurate compared to a single checkerboard approach.

In the field of visual odometry and motion estimation, Steinbrucker et al. (2011) present a system for frame-to-frame trajectory estimation by minimizing an energy function in the space of the dense depth data. Dryanovski et al. (2012) present a system for frame-to-frame registration using edge features, which uses a high-frequency loop for sparse data and a low-frequency loop for dense data. Henry et al. (2010) present a system which uses GPU-accelerated SIFT features. Images are aligned on a frame-to-frame basis, by using both sparse and dense data. Global refinement is performed offline using Sparse Bundle Adjustment. In their newer work Henry et al. (2012) they also consider FAST features without GPU acceleration.

Endres et al. (2012) present a system which uses sparse SURF, ORB, or GPU-accelerated SIFT descriptors (the choice is configurable). Images are aligned against a subset of previous frames of constant size, in order to increase robustness. The implementation requires multiple threads. Additional refinement of the trajectory can be performed offline. Newcombe et al. (2011) present a system for RGB-D pose tracking and mapping. Their method aligns dense depth data against a model of a surface. The model is augmented with new data. The system requires a GPU-equipped computer. Kerl et al. (2013) describe a RGB-D SLAM system which tracks images using a frame-to-keyframe scheme and performs pose-graph alignment in a separate thread. The system is notable for performing dense alignment between the frames, using data from all pixels. The alignment runs in real time on a desktop CPU; however, it requires more computational time and lower resolution images than our proposed algorithm, making our solution more appropriate for computationally-constrained systems. Furthermore, their work does not demonstrate results from environments larger than a limited office scene; in comparison, we provide results from a larger-scale reconstruction of multiple rooms. Meilland and Comport (2013) present a system which unifies keyframe-based SLAM techniques with volumetric map representations. The system runs in real time, and requires a GPU-equipped computer. Whelan et al. (2015) introduce a GPU based ICP for realtime pose estimation and dense map fusion algorithm called ‘ElasticFusion’. It introduces a ‘surface loop closures’ approach rather than feature based graph model loop closing. The current most successful RGB-D SLAM is proposed by Mur-Artal and Tardós (2017), it proposes an ORB feature based local and global graph optimization approach to perform pose estimation. It has achieved the best performance almost in all public dataset including TUM dataset (Sturm et al. 2012), KITTI dataset (Geiger et al. 2012), and EuRoc Dataset (Burri et al. 2016). However, ORB-SLAM2 still requires heavy graph optimization, which is not suitable for real time processing on a mobile platform with computation constraints.

In our previous work Dryanovski et al. (2013), we describe a visual odometry pipeline using a persistent feature model. The visual odometry section of this paper expands upon our previous results, describing an improved pipeline including a RANSAC step, as well as more experimental evaluations.

3 RGB-D measurement model

3.1 Preliminaries

In the context of this paper, we define an *RGB-D image* as an image with 3 standard color channels and an additional *depth* channel. The depth channel contains a measure, in meters, of the distance from the camera optical center to the scene, along the camera optical axis. We adopt the standard camera coordinate frame convention with the z axis pointing along the optical axis. We expect RGB-D images to have complete color information, and incomplete, yet very dense, depth information.

An RGB-D camera will typically output a color and depth image produced by two separate cameras, RGB and infrared, each with its own intrinsic parameters and distortion coefficients. The device and its software driver might have a built-in calibration for the intrinsics of the two cameras, as well as the extrinsic pose between them. This allows for the two images to be properly undistorted, and for the depth image to be reprojected into the frame of the RGB camera to form a single unified RGB-D image.

In the case when the factory calibration is not provided, or is not sufficient, a user can perform a custom calibration in two steps. First, the intrinsic parameters of the RGB and infra-red (IR) cameras are determined, including a radial distortion model. This is performed by observing a black and white checkerboard of known size in multiple images. For this procedure we used existing toolboxes implemented with OpenCV (Bradski 2000). The approach has been well documented, for example in Wang et al. (2010). Besides, there exists a well built open source tool based on ROS (James Bowman 2017) with detailed instruction. A point of interest is that the infrared pattern projected by the RGB-D camera interferes with the corner detection on the IR images of the checkerboard. Therefore, it is necessary to place something in front of the IR projector which diffuses the infrared light pattern (a white sheet of paper sufficed in our experiments). This allows the treatment of the IR image as a standard monochrome image.

The second step is determining the extrinsic matrix between the IR and RGB cameras. This can be performed by treating the two cameras as a stereo pair, and calibrating by using a checkerboard observed in multiple images.

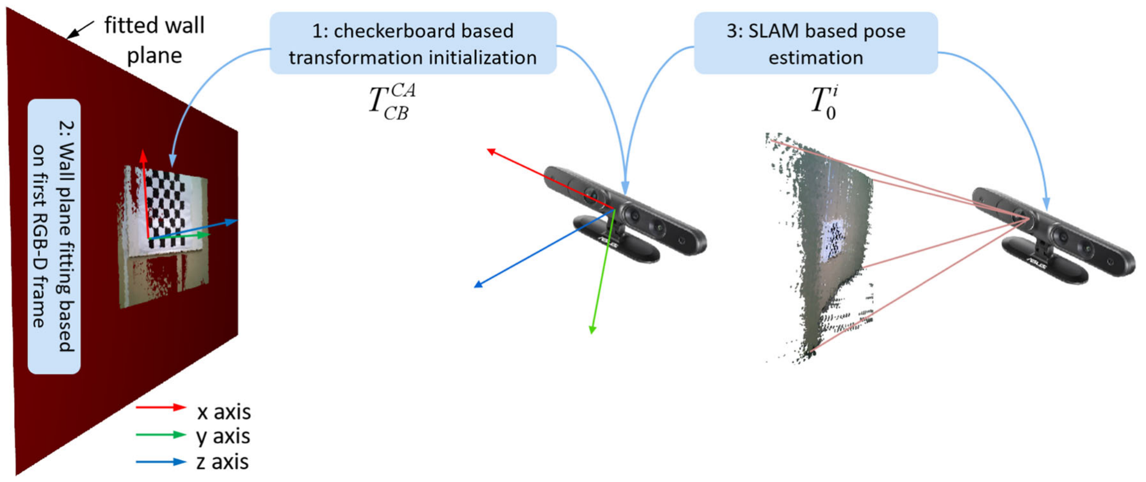

fitted wall plane
1: checkerboard based transformation initialization
T_{CB}^{CA}
3: SLAM based pose estimation
T_0^i
2: Wall plane fitting based on first RGB-D frame
x axis
y axis
z axis

Fig. 3 The depth bias calibration system setup proposed in this paper. It follows the three steps as described in Sect. 3.2. First, we use checkerboard to obtain the initial transformation from the camera to the wall plane T_{CB}^{CA} and perform wall plane fitting to obtain the depth ground truth. Then, we estimate the pose of each RGB-D frame based on ORB-SLAM2. The depth error is calculated by comparing to the wall plane

Stereo camera calibration is also a well studied problem. We use an implementation based on the OpenCV stereo calibration functions (Bradski 2000). Treating the two cameras as a stereo pair can be achieved by the diffusion of the IR image described above.

Once the RGB-D image is properly formed, we can represent the image as a dense, 3D point cloud. Given a pixel $\mathbf{q} = [U, V, Z]^T$, where U and V are the image coordinates, and Z is the measured depth, in meters, we can express $\mathbf{q}$ as a 3D point $\mathbf{p} = [X, Y, Z]^T$, in the camera coordinate frame:

$$X = \frac{Z}{f_x}(U - c_x) \tag{1a}$$

$$Y = \frac{Z}{f_y}(V - c_y) \tag{1b}$$

where f_x, f_y are the focal lengths, and c_x, c_y are optical centers of the RGB-D image, taken from the RGB-D intrinsic matrix. We treat U, V, Z as random variables. The former two reflect the uncertainty location in sparse features such as corners. The latter reflects the error in the sensor's depth measurements.

3.2 Depth bias calibration

Once we have obtained the intrinsic and extrinsic camera matrices, we can use the RGB camera as a source of reference for calibrating the depth camera. While not a real ground-truth measure, we assume that the measurement of the pose by using SLAM and checkerboard pose estimation is significantly more accurate than the depth data returned in the depth image. We note that the results of any subsequent depth calibration are subject to the quality of the extrinsic, intrinsic, and distortion model calibration of the cameras, as well as the SLAM pose estimation. To calibrate the depth bias, our method takes three steps (see in Fig. 3):

(1) *Initial pose estimation* We attach a large checkerboard on a flat wall very tightly. Then, we put the camera facing toward the wall and start recording videos. For the first frame, we perform RANSAC projective-n-points (PNP) algorithm to estimate the initial transformation between the checkerboard and the camera, e.g., T_{CB}^{CA}.
(2) *Plane fitting* Once we obtain the initial pose, T_{CB}^{CA}. We can transform the point cloud of the first frame to the checkerboard coordinate system. Then, we perform plane fitting to obtain the center ${}^{CB}P$ and the normal ${}^{CB}N$ of the wall plane.
(3) *SLAM pose estimation and depth error calculation* For each new frame, we obtain the pose using SLAM and transform to the checkerboard coordinate system. Then, we can simply calculate the error of each 3D point based on the initially estimated plane $\{({}^{CB}P, {}^{CB}N)\}$.

For performing the depth calibration, we did two separate experiments in both indoor and outdoor environments. The setup is illustrated in Fig. 3, we use a large checkerboard (both chessboard and Aruco board (Garrido-Jurado et al. 2014) are used) printed on a thin cardboard and then attach it to flat wall surfaces tightly. Since there is a limit on how large we can print the checkerboard on a flat surface, when we move the checkerboard further away from the camera it does not cover the entire field of view of the RGB image. Therefore we need

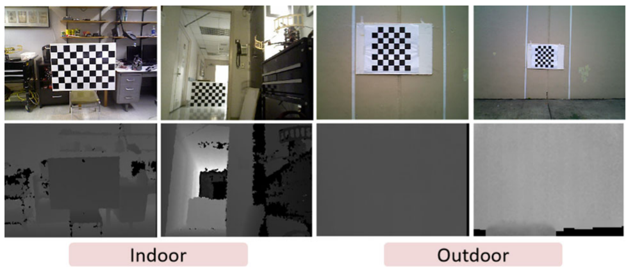


Fig. 4 Example of 2 sets of RGB-D training pairs used for the depth calibration, which are indoor and outdoor RGB-D frames respectively. The RGB-D frames are taken at different distance

to shift the checkerboard and create an image mosaic. For outdoor scenario, if a flat wall is available, we could attach the checkerboard to the wall and observe it from various distances.

To mitigate the effects of the noise in the depth readings, we collected: (1) Indoor: checkerboard with an averaging 1000 consecutive depth images, Aruco board with an averaging 2700 consecutive depth images; (2) Outdoor: checkerboard with around 5000 consecutive depth images from a large flat wall. An RGB and (average) depth image together form a training pair. Figure 4 shows an example of two training pairs.

In each training pair, we estimate the pose of the camera using our pose estimate system (in Fig. 3) in the RGB camera frame. Next, we transform them into the IR camera frame using the extrinsic matrix between the cameras. We generate a ground-truth (reference) depth image defined by the fitted wall plane $\{({}^{CB}P, {}^{CB}N)\}$. For indoor scenario, any pixels which are outside the checkerboard are not used, while the outdoor scenario keeps all the pixel on the wall plane. Thus, the indoor training image pair only allows for calibration in a small area of the image, which gets smaller as the checkerboard is placed at greater distances. Effectively, this means that we need to record a large number of training pairs in order to cover the entire field of view. Figure 5 shows an overview of the data we collected in indoor environment. On the left is a side view of multiple reference checkerboards. In the middle are the corresponding measured depth readings for each checkerboard. On the right is a closeup for the reference and measured locations for a single checkerboard, clearly showing the discrepancy between the two. Note that for clarity, the image only shows 50 checkerboard images.

Once all N training images are collected, we create a set of training points for each checkerboard pixel in each image. We denote the measured distance at a pixel $\mathbf{q} = [u, v]$ as $z_{uv}^{(i)}$, and the corresponding reference distance $\mathcal{Z}_{uv}^{(i)}$. Since the same pixel $\mathbf{q}$ will be observed at multiple images, we use i to express the index of the training image. We define a "corrected" reading $\widetilde{z}_{uv}$ described by a set of c_{0uv}, c_{1uv} and

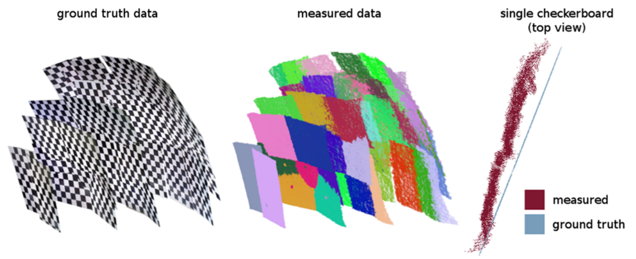


Fig. 5 Left: reference checkerboard images, detected from the RGB image using our pose estimation system. Center: corresponding *measured* checkerboards in the depth image. Image shows 50 checkerboards (173 were used for the actual calibration). Right:close up (top view) of a single reference and measured checkerboard pair, clearly showing the discrepancy between the expected and depth readings

c_{2uv} coefficients:

$$\widetilde{z}_{uv} = \sum_{i=0}^{n} c_{iuv} \cdot z_{uv}^{i} \tag{2}$$

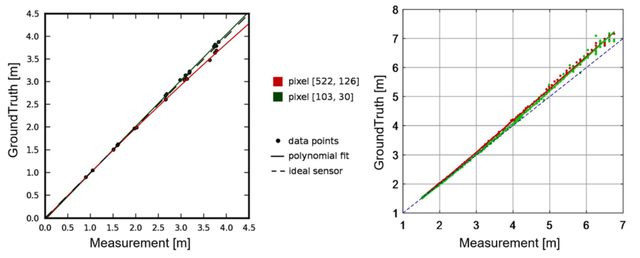
In this paper, the total error function for a given pixel location u, v can be defined as

$$e(u, v) = \sum_{i} \left(\mathcal{Z}_{uv}^{(i)} - \widetilde{z}_{uv}^{(i)} \right)^2 \tag{3}$$

where i iterates across all the images where the pixel u, v was inside the checkerboard area, c_{iuv} denotes per-pixel coefficients. We also notice the same problem which was discussed in Basso et al. (2018), that is, what should be the best polynomial order n to fit the depth error distribution over each pixel. To find the best model for depth bias fitting, we valid the options in an experimental approach, where we select the order as $n = \{2, 3, 4, 5, 6, 7, 8\}$ and then learn their polynomial coefficients respectively based on the principle,

$$\arg\min_{c_{0:2uv}} e(u, v) \tag{4}$$

We accomplish this by fitting a *n-order* polynomial to the data. The raw depth and error distribution over each pixel is illustrated in Fig. 6, we perform two pose estimation approaches to obtain the error results, which are PNP based on checkerboard and our proposed coupled pose estimation method (see Fig. 3). The figure demonstrates that some parts of the depth image consistently overestimate the depth reading, while others consistently underestimate it. Based on the ground truth we have, we believe the PNP approach introduced none-negligible depth error to the system. Again, we calculate the error based on the wall-fitting at the initial stage, and then simply calculate the distance from the point to the wall as error. To decide which order of polynomial should be chosen, we valid a total of 7 different



(a) PNP (b) Our Pose System

Fig. 6 Comparison of depth calibration results for a two different pixels in the image ([522, 126] and [103, 30]) using **a** PNP and **b** our pose estimation. The figure shows the polynomial fit (solid line) and its deviation from the ideal sensor (dashed line). In **a**, we can see that one of the pixels systematically overestimates the depth Z, while the other pixel systematically underestimates it. In **b** we can easily derive that all the pixel will under estimate if Z is over 4 m. Based on our observation, we found **a** is a wrong conclusion

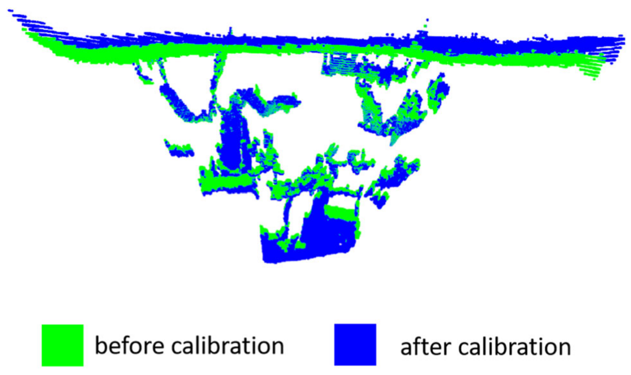


Fig. 7 Top-down view of point cloud created by observing a flat wall with furniture in front of it. The image shows the point cloud before and after applying the polynomial correction. The unwarp behavior is especially visible at upper side of the image

orders, $n = \{2, 3, 4, 5, 6, 7, 8\}$. It can be seen in Fig. 8 that second order polynomial achieves almost the same performance as the eighth order fitting. Also, depth correct can largely decrease the systematic depth error. Thus, in this paper, we choose to use the second order fitting for both accuracy and time performance considerations. Since the fit is performed for each pixel, the end result is 640×480 sets of coefficients. In our implementation, we store them as three VGA images $I_{c_0}, I_{c_1}, I_{c_2}$, each for the respective coefficient ranks.

Once the coefficient images are recovered, they can be applied to incoming depth images to "unwarp" them and remove the systematic bias in the depth reading on a per-pixel basis. The results of the unwarping for a single depth image are displayed in Fig. 7. The figure shows the point clouds reconstructed from the original depth image and the unwarped one. We also perform rooted mean square (RMS) analysis for the error between the measured distance z in the depth image and reference $\mathcal{Z}$ obtained from the RGB checkerboard image. Figure 8 presents the results for an Asus Xtion-PRO sensor. We define the RMS errors for a given the uncalibrated and calibrated image i as:

$$e_{rms}^{(i)} = \left(\frac{1}{n}\sum_{u,v}\left(\mathcal{Z}_{uv}^{(i)} - z_{uv}^{(i)}\right)^2\right)^{\frac{1}{2}} \tag{5a}$$

$$\widetilde{e}_{rms}^{(i)} = \left(\frac{1}{n}\sum_{u,v}\left(\mathcal{Z}_{uv}^{(i)} - \widetilde{z}_{uv}^{(i)}\right)^2\right)^{\frac{1}{2}} \tag{5b}$$

where u and v iterate over all the checkerboard pixels observed in that image. Each data point in the figure represents the RMS error of z over all the pixels in a given checkerboard test image, versus the average depth of the checkerboard. the results with and without calibration are

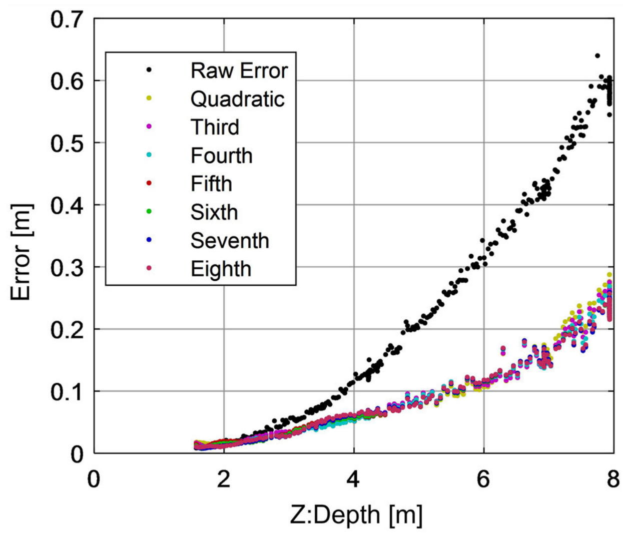


Fig. 8 Mean RMS error in Z before calibration $e_{rms}^{(k)}$ and after $\widetilde{e}_{rms}^{(k)}$, for Asus Xtion-PRO camera. Each data point represents the mean RMS error over the wall plane at varying distances Z from the camera. We illustrate depth error using second order to eighth order fitting

displayed. The figure shows that the RMS error is significantly lower when the polynomial unwarping is performed on the depth images, and the error improvement becomes more pronounced as the distance between the object and the camera grows.

We performed the same procedure with a depth camera embedded in a mobile device—Peanut, Google's Project Tango prototype cell phone (Google 2014). The depth camera, which is based on the same structured light technology as the the Asus camera, exhibited much higher initial biases (as high as 1.5 m at a distance of 3 m in some areas of the image). After calibration, we were able to reduce the RMS error by approximately an order of magnitude (Fig. 9).

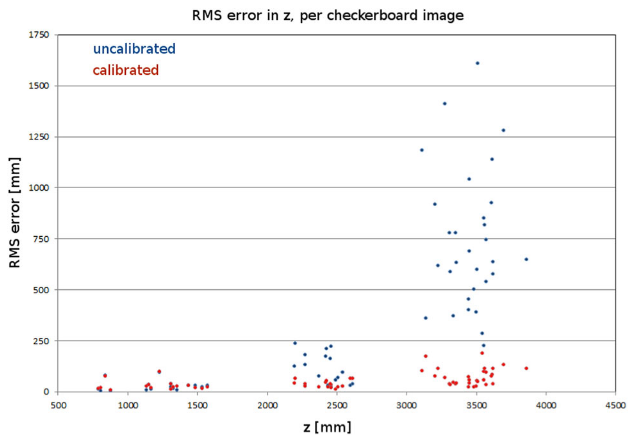


Fig. 9 RMS error in Z before calibration $e_{rms}^{(k)}$ and after $\widetilde{e}_{rms}^{(k)}$, for depth camera on mobile device. Each data point represents the error over the checkerboard pixels in the i-the checkerboard image. Multiple checkerboard images at varying distances Z from the camera are used

3.3 Depth uncertainty analysis

Khoshelham and Elberink (2012) present the following formulation for the uncertainty in Z:

$$\sigma_z = \frac{1}{fb}\sigma_d \mu_z^2 \tag{6}$$

where f is the effective IR camera focal length, and b is the baseline distance between the IR camera and IR projector. After calibration, Khoshelham and Elberink (2012) obtain the following expression:

$$\sigma_z = 1.45 \times 10^{-3}\mu_z^2 \tag{7}$$

where μ_z and σ_z are expressed in meters.

We extend the model in two ways. First, the locations of features detected by a sparse feature detector are subject to error; therefore, we allow for uncertainty in the U and V variables. Second, we assume that the depth uncertainty is dependent not only on the depth readings of a given pixel, but also on its neighbors in a local window. We will show that using these assumptions, we can predict the magnitude of the depth uncertainty better than the previously published model.

Let us begin by assuming that U and V are independent random variables distributed according to a normal distribution $\mathcal{N}(\mu_u, \sigma_u)$ and $\mathcal{N}(\mu_v, \sigma_v)$ respectively. Further, let σ_u and σ_v inform the following approximate Gaussian kernel of size 3×3:

$$W = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{8}$$

Assuming that Z is normally distributed, we can define a random variable $\widehat{Z}$, which is a mixture of the Z variables in a local window $\{i :\in [\mu_u - 1, \mu_u + 1], j \in [\mu_v - 1, \mu_v + 1]\}$. The weights of the mixture w_{ij} are chosen according to the kernel W. The mean and variance of the resulting Gaussian mixture $\widehat{Z}$ are

$$\widehat{\mu}_z = \sum_{i,j} w_{ij}\left(\mu_{z_{ij}}\right) \tag{9a}$$

$$\widehat{\sigma}_z^2 = \sum_{i,j} w_{ij}\left(\sigma_{z_{ij}}^2 + \mu_{z_{ij}}^2\right) - \widehat{\mu}_z^2 \tag{9b}$$

At this stage, we have two alternative models for the uncertainty: σ_z, estimated according to the simple model in Eq. 6, and $\widehat{\sigma}_z$, estimated according to the Gaussian mixture model. To evaluate which model has more predictive power, we gather n depth images of a static scene. For each pixel in the image, we calculate the uncertainty $\bar{\sigma}_z$ in the metric depth z according to

$$\bar{\mu}_z = \frac{1}{n}\sum_{m=1}^{n} z_m \tag{10a}$$

$$\bar{\sigma}_z^2 = \frac{1}{n-1}\sum_{m=1}^{n} (\bar{\mu}_z - z_m)^2 \tag{10b}$$

We call this the *observed* uncertainty. When n is large (we used 200 images) we can assume that the observed uncertainty approaches the true uncertainty for the RGB-D camera measurement. Next, we take a single depth image, and generate the two *predicted* uncertainties σ_z and $\widehat{\sigma}_z$. Figure 10 shows a comparison between the observed and predicted uncertainties according to both models. We note that the Gaussian mixture model predicts the uncertainty much better than the simple model, especially around the edges of objects. This comes from the fact that the RGB-D camera produces readings for edge pixels, which tend to jump from foreground to background.

3.4 3D distribution

We can estimate the 3D uncertainty Σ of a point $\mathbf{p}$ from Eq. 1.

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 \end{bmatrix} \tag{11}$$

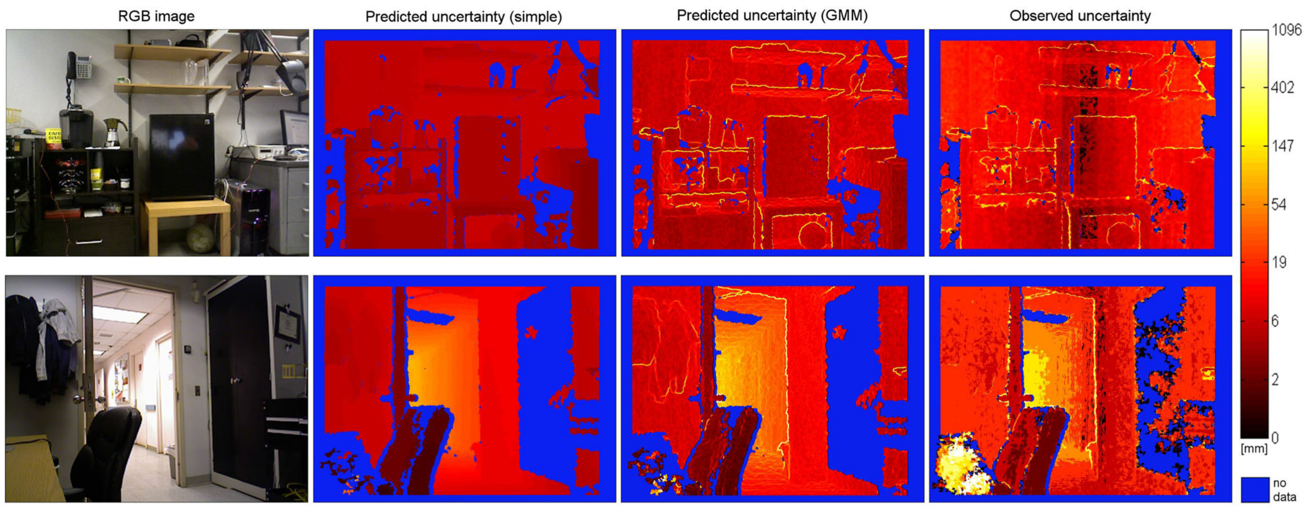


Fig. 10 Uncertainty analysis for the depth readings of an RGB-D camera. Left: RGB image, shown for visualization only. Right: *observed* (ground truth) uncertainty ($\bar{\sigma}_z$), obtained by taking 200 depth images of a static scene and calculating the mean and standard deviation on a per-pixel basis. Color is scaled as the log of the uncertainty. Center-left: uncertainty predicted from a single depth image, according to the simple model (σ_z). Center-right: uncertainty predicted from the same depth image, according to the Gaussian mixture model ($\widehat{\sigma}_z$). We demonstrate the Gaussian model uncertainty predicts the true uncertainty more accurately, especially around object edges

where

$$\sigma_x^2 = \frac{\sigma_z^2(\mu_u - c_x)(\mu_v - c_y) + \sigma_u^2(\mu_z^2 + \sigma_z^2)}{f_x^2}$$
$$\sigma_y^2 = \frac{\sigma_z^2(\mu_u - c_x)(\mu_v - c_y) + \sigma_v^2(\mu_z^2 + \sigma_z^2)}{f_y^2}$$
$$\sigma_{xz} = \sigma_{zx} = \frac{\sigma_z^2(\mu_u - c_x)}{f_x}$$
$$\sigma_{yz} = \sigma_{zy} = \frac{\sigma_z^2(\mu_v - c_y)}{f_y}$$
$$\sigma_{xy} = \sigma_{xy} = \frac{\sigma_z^2(\mu_u - c_x)(\mu_v - c_y)}{f_x f_y}$$

The above expressions are derived with μ_z and σ_z from the simple depth uncertainty model. We can then approximate Σ in terms of the Gaussian mixture model by replacing μ_z and σ_z with $\widehat{\mu}_z$ and $\widehat{\sigma}_z$ respectively. We approximate **p** as a multivariate Gaussian distribution with mean $\boldsymbol{\mu} = [\mu_x, \mu_y, \mu_z]^T$ and covariance Σ.

4 Trajectory estimation

4.1 Overview

The entire visual odometry pipeline is shown in Fig. 11. The trajectory estimation takes four steps as illustrated in the figure: (1) firstly, feature detection over the intensity image; (2) secondly, 3D back-projection and covariance calculation based on Sect. 3.3; (3) thirdly, 3D feature point cloud ('model') based ICP pose estimation; (4) finally, update the feature point cloud using Kalman filter. We take advantage of the filter-based approach to update the feature point cloud and use the feature point cloud to estimate the current pose. Thus, the current frame pose estimation does not only relies on the last frame but also relies on all previous frames that captured the same features.

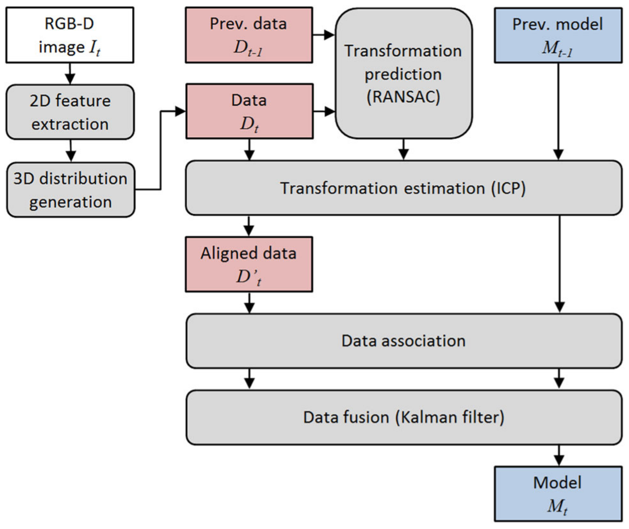


Fig. 11 Pipeline for the trajectory estimation. We align sparse feature data from the current RGB-D frame to a global *model*. The data is represented by 3D points with covariance matrices

The estimation begins with extracting sparse features in each incoming RGB-D image I_t. The features are detected on the intensity channel of the RGB image. We have experimented with several choices of feature detectors, including SURF (Bay et al. 2008), ORB (Rublee et al. 2011), and Shi and Tomasi (1994) keypoints. While our implementation offers a configurable choice between them, we found that the Shi-Tomasi features offer the best trade-off between robustness and computational speed. Besides, we spatially split an intensity image into 8 patches, and force the feature detector to detect the same number of features in each patch. It is able to provide a more accurate pose estimation.

Next, we back-project the feature to 3D space and estimate the 3D normal distribution for each feature, according to the uncertainty equations defined in Sect. 3. From this, we generate a set of 3D features $D = \{\mathbf{d}_i\}$. Each feature $\mathbf{d} = \{\boldsymbol{\mu}^{[D]}, \Sigma^{[D]}\}$ has a mean and covariance matrix. The D set is expressed in the camera reference frame.

Meanwhile, we initialize a global feature point cloud $M = \{\mathbf{m}_j\}$, with $\mathbf{m} = \{\boldsymbol{\mu}^{[M]}, \Sigma^{[M]}\}$, expressed in the world coordinate frame. It is called 'model' in this paper, and it is used to perform a 3D (model) to 3D (current frame) Iterative Closest Points (ICP) (Besl and McKay 1992) based pose estimation. In contrast, the frame-to-frame based pose estimation only relies on the last frame (Mur-Artal and Tardós 2017), and does not take advantage of the historical observations. Thus, it is easy to drift without using bundle adjustment. Our proposed 'model' which integrates the historical observations is able to provide a more accurate pose estimation.

Then, the trajectory estimation is basically the process of aligning the current feature point cloud D to the global *model* M. This happens in two steps. *First*, we align D_t to D_{t-1}, where t is time step, using a 3-point RANSAC (Fischler and Bolles 1981) transformation estimation. *Second*, we align D_t to the previous model M_{t-1}, using the transformation estimated by RANSAC as a initialization to do ICP. *Note* that we do not use the associations generated by RANSAC to do ICP, but choose to compute them all over iteratively using the nearest neighbor search. This is because the associations from RANSAC are only between the current and last frame, while in the ICP step we are interested in aligning the 3D features to the global model M, which potentially contains features not seen in the previous frame. While this might seem like a duplicated effort, in practice the *model* M keeps updated consistently thus a single or few abnormal pose estimation will not affect the robustness of the overall performance.

Once the final transformation is found using the above two-step approach, we transform the set D_t into the global frame (expressed as D'_t). Finally, we establish the final correspondences between the two sets D'_t and M_{t-1}. Re-observed features in the 'model' M_{t-1} are updated using a Kalman filter, and new features are added to the model M_{t-1}, resulting in the new model M_t. The rest of this section describes the details of the third and fourth steps on how to perform pose estimation and model update.

4.2 Model registration

This section discusses the pose estimation that uses the two-step approach as discussed in Sect. 4.1. We first use RANSAC to estimate the initial guess of the current frame for the first step. It involves with a feature matching between the current frame and last frame. For the second step, we begin with defining a distance function *dist* which measures the distance between two features $\mathbf{f}_a$, $\mathbf{f}_b$. Each feature is normally distributed with means $\boldsymbol{\mu}_a$ and $\boldsymbol{\mu}_b$ and covariance matrices Σ_a and Σ_b.

$$dist(\mathbf{f}_a, \mathbf{f}_b) = \sqrt{\Delta_{\mathbf{f_a f_b}} (\Sigma_a + \Sigma_b)^{-1} \Delta^T_{\mathbf{f_a f_b}}} \tag{12}$$

where

$$\Delta_{\mathbf{f_a f_b}} = \boldsymbol{\mu}_a - \boldsymbol{\mu}_b \tag{13}$$

The distance function is based on the Mahalanobis distance from a point to a distribution.

Then, we use ICP to align the data set D to the model set M. The ICP algorithm has 2 steps of interest: (1) generating correspondences between the two input sets, and (2) calculating the transformation which minimizes the distance between the correspondences. In a classical ICP formulation, the correspondences are generated using nearest neighbors in Euclidean space; the transformation is also estimated by minimizing the sum of squared Euclidean distances.

In this paper, we propose using a modified ICP algorithm, in which we establish approximate correspondences using the Mahalanobis distance. First, we build a kd-tree Muja and Lowe (2009) of the model M, by using the means of the features. Next, for each feature $\mathbf{d}$ in the data D we find the k nearest Euclidean neighbors from M. Finally, we iterate through all k candidates, and find the one which has the smallest Mahalanobis distance. This allows us to leverage the efficiency of kd-trees, which cannot be directly used with non-linear functions such as the Mahalanobis distance. In our implementation, we use a small size for k (for example, 4).

The rest of the ICP algorithm remains the same. We note that while it is possible to optimize a Mahalanobis distance as the objective function for the best transform, we do not do so, and this is a possible area of improvement. An example of an algorithm which implements a similar optimization (albeit in the context of dense data) is Generalized ICP (Segal et al. 2009).

4.3 Data association and model updating

This section addresses the model updating, which consistently fuses the historical observations to the global model. For model updating, we begin by data association step via rotating the data set D into the global frame of reference, and refer to it as $D^{'}$. Let the current transformation between the global and camera coordinate frames is T, consisting of a rotation and translation:

$$T = \begin{bmatrix} \mathrm{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{14}$$

We can transform the mean vector and covariance matrix according to:

$$\boldsymbol{\mu}^{'} = \mathrm{R}\boldsymbol{\mu} + \mathbf{t} \tag{15a}$$

$$\Sigma^{'} = \mathrm{R}\Sigma\mathrm{R}^{T} \tag{15b}$$

Next, for each point $\mathbf{d}_i^{'}$ in the transformed *Data* set, we find the approximate nearest Mahalanobis neighbor $\mathbf{m}_j$ in the *Model*.

$$dist(\mathbf{d}^{'}, \mathbf{m}) = \sqrt{\Delta_{\mathbf{d}'\mathbf{m}} (\Sigma^{[M]} + \Sigma^{[D^{'}]})^{-1} \Delta_{\mathbf{d}'\mathbf{m}}^{T}}$$

We consider two points to be associated if the distance between them is lower than a threshold ϵ. Typical values for ϵ include 7.82 or 11.35. The two thresholds correspond to the 95% and 99% probability tests that the data point is sampled from the given model distribution.

Any features in $D^{'}$ which cannot be associated are inserted as new members in M. The model is bounded in size, so if the maximum allowed size is exceeded, we remove the oldest features in the model. This is achieved by using a ring-buffer implementation.

For each feature which is associated, we perform a Kalman filter update. We treat the distributions $\mathbf{m}$ as the prior, and the distribution $\mathbf{d}$ as the observation.

Since we do not have a prediction in our *model*, it keeps the same as the prior state of the model at time $t-1$, that is,

$$\tilde{\boldsymbol{\mu}}_t = \boldsymbol{\mu}_{t-1}^{[M]} \tag{16a}$$

$$\tilde{\Sigma}_t = \Sigma_{t-1}^{[M]}. \tag{16b}$$

Then, the new observations are used to update the associated feature states at time t by applying the following equations,

$$G_t = \tilde{\Sigma}_t \left(\tilde{\Sigma}_t + \Sigma_t^{[D]} \right)^{-1} \tag{17a}$$

$$\boldsymbol{\mu}_t^{[M]} = \tilde{\boldsymbol{\mu}}_t + G_t \left(\boldsymbol{\mu}_t^{[D]} - \tilde{\boldsymbol{\mu}}_t \right) \tag{17b}$$

$$\Sigma_t^{[M]} = (I - G_t)\, \tilde{\Sigma}_t \tag{17c}$$

where G is the kalman gain, $\boldsymbol{\mu}_t^{[M]}$ is the global associated global feature points, $\boldsymbol{\mu}_t^{[D]}$ is the current observation.

5 Post Bundle adjustment and reconstruction

This section discusses the post-processing towards global Bundle adjustment to achieve 3D structure reconstruction. We also adopt using keyframes to represent and reconstruct a scene (Klein and Murray 2007). New keyframes are generated heuristically, using either distance or overlap metric. The distance metric triggers a new keyframe when the linear or angular distance traveled between the current camera pose (as reported by the visual odometry) and the camera pose of the last keyframe exceeds a certain threshold (for example, 0.3 m or 30°). The overlap metric triggers a new keyframe when the number of features in the data set D which have a correspondence in the model M falls under a certain threshold.

Each of the RGB-D keyframes informs a vertex in a graph. The vertex is described by the keyframe's 6-DoF pose. Edges in the graph are described by the relative pose between two keyframes. The visual odometry pipeline provides the edge information for time-consecutive keyframes. To perform large-scale loop closure and global alignment, we need to detect pairs of non-consecutive keyframes and a relative pose between them. Once the graph is built, we optimize for the pose of all keyframes using a non-linear graph optimizer (g^2o, Kummerle et al. 2011). Currently, the pose graph adjustment does not modify the location of the features in the model used for trajectory estimation. However, since we perform the alignment as the final step at the end of the reconstruction run, this is not an issue.

The main problem in the approach described above is finding pairs of RGB-D keyframes that observe the same scene, so we can calculate their relative pose. We refer to this as the *place recognition* problem. Our general approach to the place recognition problem is the following: select a pair of images, try to align their 3D features using a sample-and-consensus algorithm, and mark them as correlated if the algorithm finds a transformation model with a large number of inlier features. We present two ways to implement this approach: a *brute-force* implementation, which is deterministic, and considers all possible image pairs and transformation models, and a *heuristic* implementation, which uses non-deterministic, randomized algorithms to find corresponding

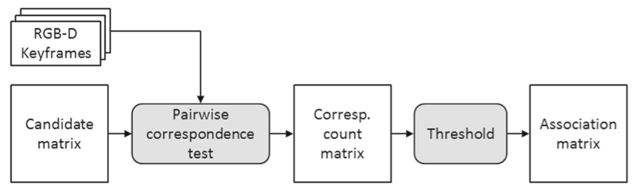


Fig. 12 Place recognition pipeline, showing the binary candidate matrix (**Q**), correspondence count matrix (**C**), and binary association matrix (**A**). The method for building **Q** and applying the correspondence test are different in the brute-force and heuristic algorithms we present

images and transformations. The former approach is prohibitively computationally expensive, but it's *complete*, so we use it as a bench-mark to validate the performance of our heuristic approach, which is computationally efficient.

The generalized form of both approaches can be seen in Fig. 12. We have a set of keyframes K with size k. First, we create a binary candidacy matrix $\mathbf{Q}$ of size $k \times k$. For any element $\mathbf{Q}_{ij} = 1$ we perform a pairwise matching test between K_i and K_j. The number of corresponding matches is stored as $\mathbf{C}_{ij}$, forming the correspondence matrix $\mathbf{C}$. Finally, we apply a threshold of minimum correspondence count on $\mathbf{C}$ to obtain the binary association matrix $\mathbf{A}$. The details of how the correspondence matrix $\mathbf{C}$ is created and what the pairwise matching test consists of are described below.

5.1 Brute-force place recognition

In the brute force place recognition algorithm, the candidacy matrix $\mathbf{Q}$ has 1 in every entry. For the pairwise matching test, we developed an algorithm called *ExSAC*, or Exhaustive Sample And Consensus. ExSAC is a deterministic version of RANSAC which considers every possible set of (fixed-size) samples to find the best model. In our case, the minimum sample size to determine the 6DoF transformation is 3 features. Thus, we use 3-point ExSAC, which considers all combinations of 3-point correspondences between a pair of RGB-D images. For each 3-point set, a rigid transformation is computed, and the number of features which are inliers to this model are counted. The output of the algorithm is the size of the best model, and the corresponding transformation. If we account for the symmetry of the test, and exclude reflexive testing, the pairwise matching is performed in total $(k-1)^2/2$ times, or $O(k^2)$. The size of the best model is stored in the correspondence count matrix $\mathbf{C}$, which is then thresholded to obtain the association matrix $\mathbf{A}$.

5.2 Heuristic place recognition

In the heuristic place recognition algorithm, we first compute a match-count matrix $\mathbf{M}$. An entry $\mathbf{M}_{ij}$ corresponds to the number of features in K_i which have their top nearest neighbors (in feature descriptor space) present in K_j. Next, for each index i, we consider the N top-scoring keyframes, in terms of match count, and mark them as 1 in the candidacy matrix $\mathbf{Q}$. The result is that $\mathbf{Q}$ has at most Nk candidates. For each candidate, we apply the 3-point RANSAC test to determine the number of correspondences. From there, the algorithm is the same as its brute-force counterpart.

The advantage of the heuristic test comes from the fact that the correspondence test needs to be performed $O(kN)$ times, with $N << k$. Furthermore, since the correspondence test uses random sampling instead of exhaustive sampling, we can use much fewer iterations to find a model which approximates the best optimal model. The main free parameter in the heuristic approach is N, or the number of top candidates to consider.

The results for a dataset of 250 keyframes can be seen in Fig. 13. On the left is the correspondence count matrix $\mathbf{C}$ for the brute force test, and the respective association matrix $\mathbf{A}$. On the right is the candidacy matrix $\mathbf{Q}$ using the heuristic approach, and the respective association matrix $\mathbf{A}$. The heuristic association matrix is nearly identical to the brute-force association matrix, validating the approach. We further analyze the recognition rate of the heuristic approach, using the brute-force as a baseline for comparison. We define the recognition rate as

$$1 - \frac{\text{false negatives}}{\text{total associations}} \tag{18}$$

where "Total associations" is the number of keyframe pair associations established by the brute force approach, and "False negatives" is the number of associations which were missed by the heuristic approach. The results for different number of candidates N are shown in Table 1.

6 Experiments

In this section, we evaluate our VO system in three aspects and compare with other state-of-the-art methods in both depth correction and localization area, using the results generated by our mobile platform which only has an Atom 1.6 GHz processor with only 2 Gb RAM. Our open-source code includes all the methods discussed in this paper to facilitate other researchers to repeat the experiments.

6.1 Model versus frame-to-frame based ICP

In the first experiment, we perform a qualitative evaluation of our visual odometry pipeline with RGB-D data recorded in an indoor environment with no ground-truth data. The camera is moved along a loop, and placed back at its starting point. We replicate the data 5 times to simulate the exact same loop. Figure 14 shows the trajectories generated using our persis-

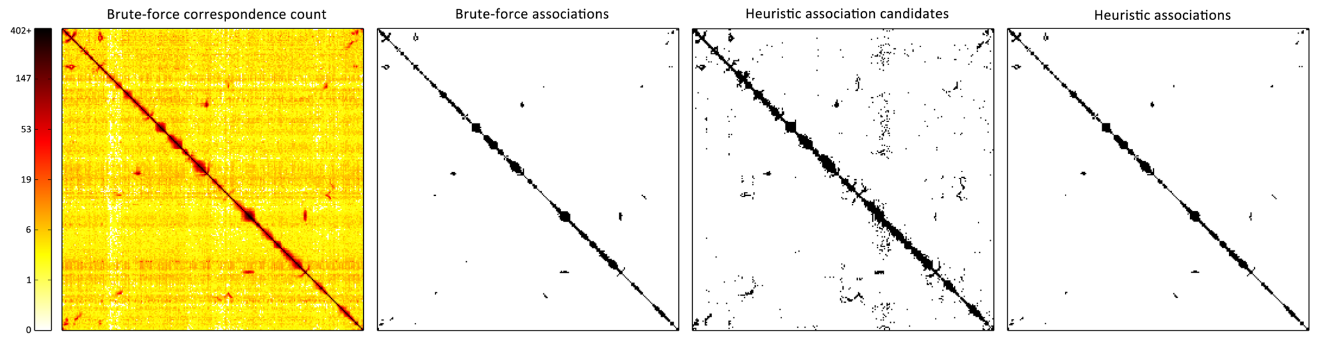


Fig. 13 Left to right: (1) correspondence count matrix **C** for the brute force test (log scale). (2) Respective brute-force association matrix **A**. (3) Candidacy matrix **Q** using the heuristic approach, showing top $N = 10$ candidates. (4) Respective heuristic association matrix **A**

Table 1 Place recognition rates

N candidates	False negatives	Recognition rate
5	169 out of 887	0.809
10	35 out of 887	0.961
15	26 out of 887	0.971
20	19 out of 887	0.979

tent model pipeline (left), versus trajectories generated by frame-to-frame ICP (right). We show that our approach is able to correctly solve small loops, keeping the trajectory error bounded. Figure 15 shows the respective sizes of the model and data feature sets. The model set grows in size during first loop, while the environment is observed for the first time. In subsequent loops, very few new features are added to the model, since most of them are correctly reobserved and associated.

6.2 Benefits of depth correction

In the second set of experiments, we discuss the benefits of the depth calibration from two perspectives, which are, trajectory accuracy comparison and 3D reconstruction error comparison. Since there is no such publicly available dataset for depth calibration performance evaluation, we create a new data set based on the TUM RGB-D dataset (Sturm et al. 2012). Although the TUM RGB-D has already provided a sequence for depth verification, the sequence only covers the lower part of the image if the distance between the camera and the chessboard exceeds 2 m. It is impossible to obtain the pixel-wise depth calibration parameters beyond this distance.

In order to obtain a dataset containing both ‘ground truth’ depth and ‘biased’ depth, we employ our depth calibration parameters (for example, pixel [320, 240] parameter illustrated in Table 2) over the ‘ground truth’ depth images, where the ‘ground truth’ is the raw depth image from the TUM dataset. Since our model is a second-order polynomial equation, we employ this to the ground truth depth in an inverse

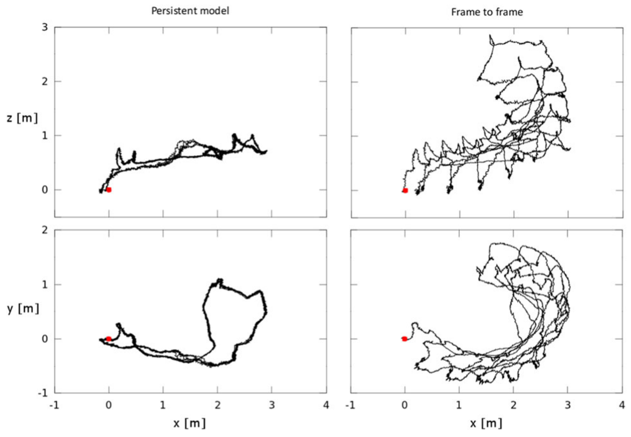


Fig. 14 Comparison of trajectory estimation with persistent model (left) versus frame-to-frame ICP. Top row: side views, xz-plane. Bottom row: top view, xy-plane. The trajectory shown consists of 5 repeated loops, with approximately 2000 images processed in each loop

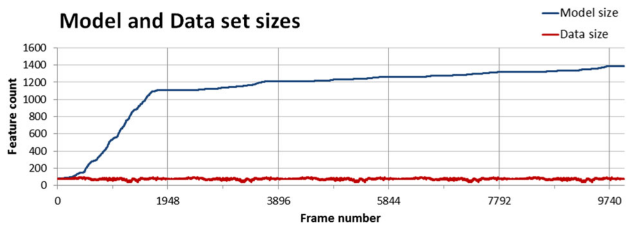


Fig. 15 Size of the model and data set sizes for the experiment in Fig. 14

Table 2 The depth calibration parameters of pixel [320, 240]

Second order	First order	Constant
$c_0^2 = 4.76e{-}03$	$c_0^1 = 9.65e{-}01$	$c_0^0 = 1.0$

approach. In Table 2, $ci_0,\ i = 0, 1, 2$ denotes parameter of each order.

We first compare the absolute trajectory error (ATE) in two cases by running ORB-SLAM2 on our GPU platform (12 core I7 CPU with 64 GB memory), with and without depth

Table 3 ATE (/m) comparison between with and without depth calibration

Dataset	ATE with depth correction	ATE without depth correction
fr1_xyz	0.0126	0.0135
fr1_rpy	0.0243	0.0264
fr1_360	0.1408	0.1248
fr1_desk	0.0206	0.0229
fr1_desk2	0.0419	0.0378
fr1_floor	0.0353	0.0490
fr2_rpy	0.0104	0.0108
fr2_desk	0.0136	0.0139
fr2_360_kidnap	0.0773	0.0795

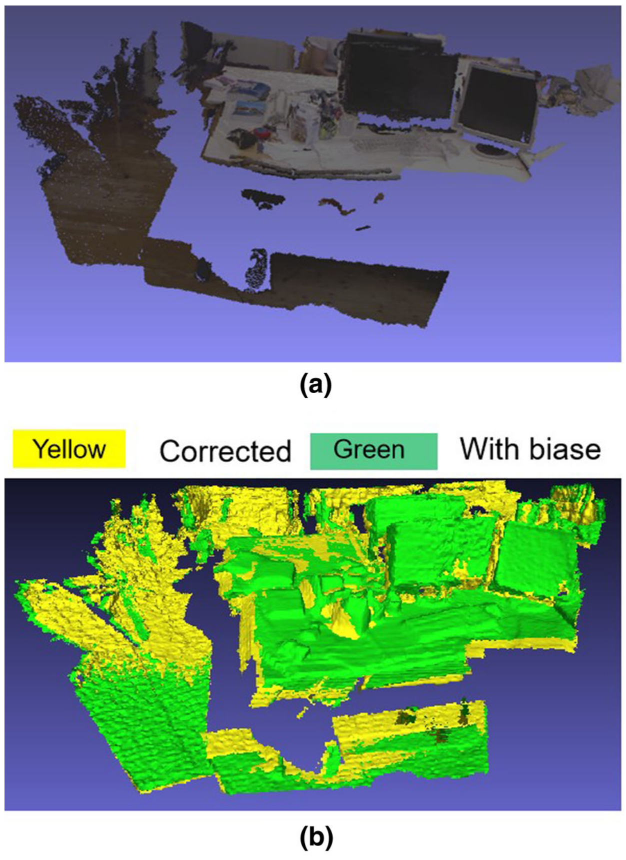


Fig. 16 Qualitative mesh comparison with/without depth calibration. **a** The color mesh of fr1_xyz dataset. **b** The mesh comparison between the depth corrected mesh (yellow) and the raw depth mesh (green) (Color figure online)

correction. For ATE evaluation, we use the open-source tool that provided by the TUM RGBD data set, to evaluate the trajectory. The result is illustrated in Table 3. It shows that the ATE is smaller after depth correction with an average 8.74% improvement. We also notice that we have slightly degraded performance in $fr1\_360$ and $fr1\_desk2$ which is caused by pose drift.

We also compare 3D reconstruction accuracy and completeness in two cases, with and without depth correction (a figure showing the mesh difference with/without using depth correction is illustrated in Fig. 16). In this paper, we choose 3D mesh accuracy and completeness as a metric to compare the reconstructed mesh (Jensen et al. 2014). For mesh accuracy, it measures the distance from customer generated mesh to the ground truth mesh, and completeness defines the distance from the ground truth mesh to the customer generated mesh. In this paper, we use a publicly accessible tool provided by Stutz and Geiger (2018). In order to obtain the result in a limited time, we sub-sample each mesh to 300,000 vertex. The results can be seen in Table 4. The table demonstrates that the mesh reconstructed with depth correction can achieve higher accuracy as well as completeness.

6.3 VO time performance comparison

In the third set of experiments, we evaluate the accuracy of the trajectory estimation using publicly-available RGB-D datasets with ground-truth trajectory information from a motion-capture camera system (Sturm et al. 2012). Table 5 shows the Relative Pose Error (RPE) for a number of different trajectories in indoor settings. We have chosen the RPE metric in order to evaluate the effectiveness of our visual odometry system. In the experiment evaluating the RPE error, we do not use our place-recognition or pose-graph-based alignment. The table compares error of our implementation (ccny_rgbd) with an existing implementation of Endres et al. (2012) (RGBD-SLAM) and ORB-SLAM2 (Mur-Artal and Tardós 2017). The results show that our implementation performs slightly better on most datasets (approximately 1cm more error), slightly worse on one dataset (3 cm less error), and significantly worse on one dataset (3 cm more error) compared with RGBD-SLAM. Our implementation performs slightly worse on most dataset (3 cm less error), and significantly worse on 1 dataset (3 cm more error) compared with ORB-SLAM2. Besides, Mur-Artal and Tardós (2017)

Table 4 3D structure accuracy and completeness comparison (/m)

Dataset	Without/with depth correction	Mesh accuracy	Mesh completeness
fr1_xyz	Without	0.0300	0.0295
fr1_xyz	With	0.0172	0.0121
fr1_360	Without	0.2456	0.2714
fr1_360	With	0.2393	0.2520
fr1_room	Without	0.0363	0.0405
fr1_room	With	0.0340	0.0327
fr1_floor	Without	0.0397	0.0415
fr1_floor	With	0.0211	0.0210
fr1_rpy	Without	0.1012	0.1164
fr1_rpy	With	0.0893	0.0971

Table 5 Relative pose error (meters)

Dataset	RGBD SLAM	ORB-SLAM2	ccny_rgbd
fr1_360	0.089	0.045	0.121
fr1_desk	0.033	0.026	0.043
fr1_desk2	0.054	0.039	0.067
fr1_room	0.095	0.035	0.068
fr1_rpy	0.046	0.039	0.064
fr1_xyz	0.021	0.019	0.030
fr2_desk	0.017	0.024	0.023
fr2_rpy	0.010	0.008	0.023
fr2_xyz	0.006	0.016	0.008
fr3_loh	na	0.021	0.028

fr1, freiburg1; fr2, freiburg2 in TUM dataset

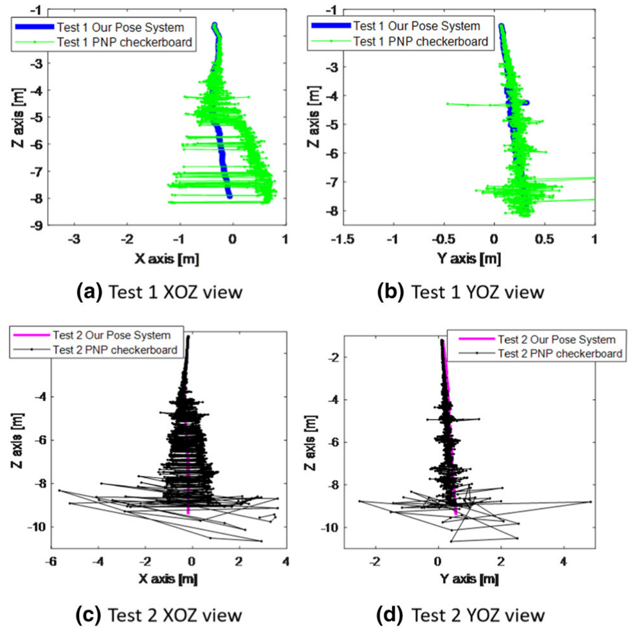


Fig. 17 The trajectory comparison of two tests in outdoor environment. We put the RGB-D camera on a moving Cart with a level. The trajectory is close to a straight line. **a** and **b** denote the XOZ view and YOZ view of the trajectory obtained from our system and PNP approach of test 1. **c** and **d** are the results of test 2

provide detailed comparison with Dense Visual Odometry (Kerl et al. 2013), and Elastic-fusion (Henry et al. 2010). Results indicates that ORB-SLAM2 outperform all these algorithms. Since our implementation is slightly worse than ORB-SLAM2 and achieve comparable performance in most cases, that is, our implementation can achieve comparable localization performance with Dense Visual Odometry and Elastic-fusion (Fig. 17).

When comparing the results, we emphasize that our implementation runs in real time (30 Hz) on VGA data using a single core of a desktop computer. Moreover, the processing time for each image is on average 22 ms, and almost never exceeds 33 ms, resulting in minimal latency (see Fig. 18). We were able to obtain similar results on an Atom 1.6 GHz processor using QVGA resolutions. Using the same setup and default parametrization for the RGBD-SLAM implementation, we were able to obtain frequencies of 5–10 Hz on the desktop machine, and unable to run in real time on the Atom computer. Besides, we also compared the time performance with ORB-SLAM2, which are illustrated in Fig. 19. ORB-SLAM2 applies local and global loop closing toward online pose estimation, however, as we can notice that it is still not suitable for real time processing on computationally constrained platform. The average processing time for ORB-SLAM2 is over 100 ms (see in Fig. 19a, c), which can only provide a 7 Hz output as we detected using *ROS Frequency Check*. For our VO implementation, it produces a much better performance with approximate 40 Hz output, that enables real time pose estimation for robot navigation application.

Thus, we believe our pipeline offers a very computationally efficient solution at a small cost of accuracy. This trade-off is especially important for systems which require real-time perception such as Micro-air vehicle flight which motivate our research. Figure 20 shows the trajectory of the visual odometry (VO) pipeline for the freiburg2_desk

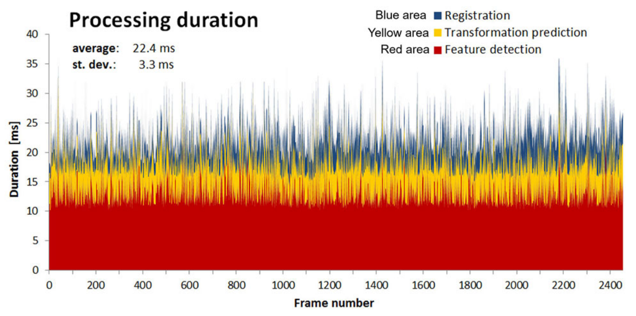


Fig. 18 Top: processing duration for each incoming image. Horizontal axis:image index number. Vertical axis: time required for feature detection (red) and alignment and model update (blue). Bottom: size of the *Data* and *Model* sets. Vertical lines mark the start of each repeated loop (Color figure online)

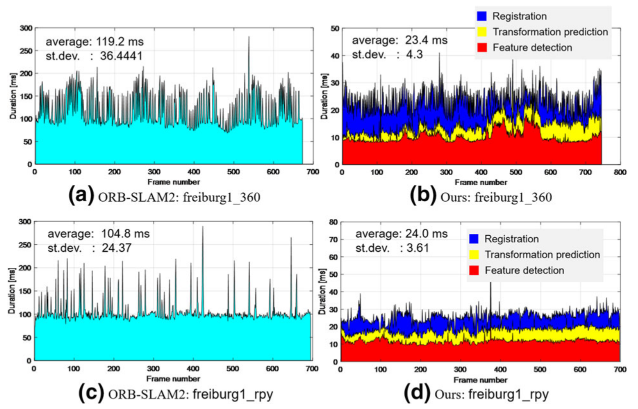


Fig. 19 Time performance comparisons between our VO and ORB-SLAM2. Where **a** and **c** show processing time of ORB-SLAM on TUM dataset *freiburg1_360* and *freiburg1_rpy*. **b** and **d** are the processing time of our VO, where we also illustrate the feature detection, pose estimation, and feature registration time of our VO

dataset. The figure also shows the trajectory after the pose-graph optimization, which was performed offline at the end of the experiment.

In the fourth experiment, we evaluated the effectiveness of the entire system, including the depth calibration, trajectory estimation, and pose-graph-based global alignment. We performed a large-scale indoor mapping experiment. We used an Asus Xtion-PRO camera carried by hand, exploring three rooms and returning to the original position in the first room. The results of the experiment can be seen in Fig. 1.

6.4 Pose system and depth correction toward accuracy

Finally, we explore how depth bias calibration affects the different modules in the system and also how our pose estimation system outperforms the single checkerboard approach. The result is illustrated in Fig. 17, we can clearly find out that pose estimation using a single checkerboard and PNP

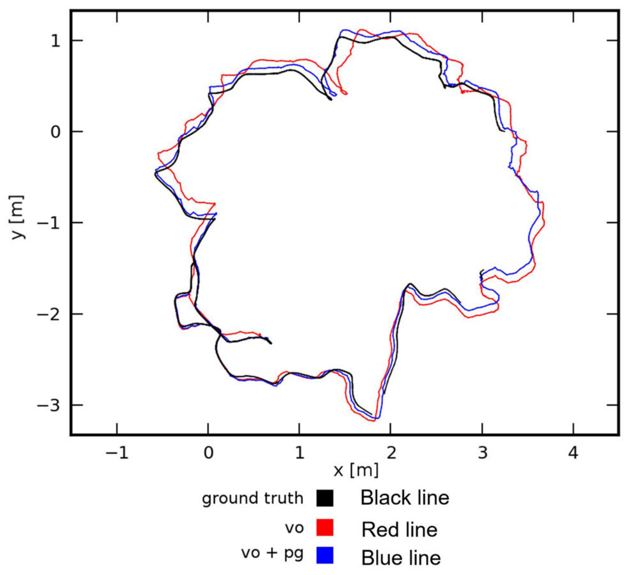


Fig. 20 Comparison of trajectories for the freiburg2_desk dataset. Black: ground truth from motion-capture camera system. Red: trajectory from our visual odometry pipeline. Blue: trajectory from our visual odometry pipeline, after pose-graph refinement (Color figure online)

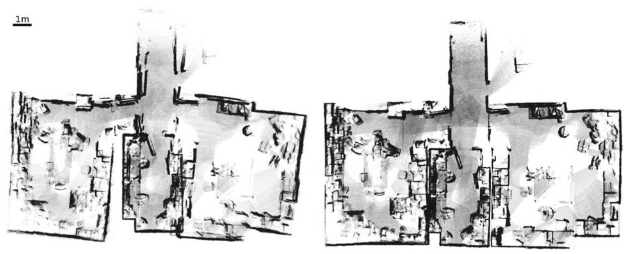


Fig. 21 Results from a large-scale 3D mapping experiment with an Asus Xtion-PRO camera. Left: orthographic projection of the 3D map, created using data without polynomial depth calibration. Right: result from same experiment, repeated with polynomial depth image correction. Both maps are generated using our visual odometry and pose-graph correction

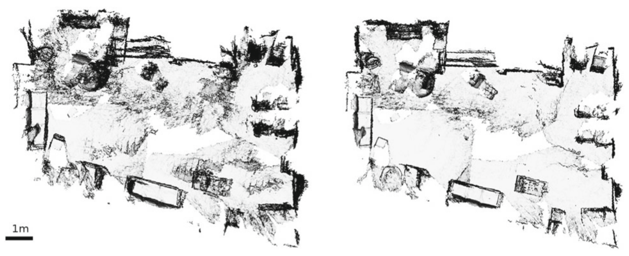


Fig. 22 Results from a 3D mapping experiment with a Google Peanut mobile device. Left: orthographic projection of the 3D map, created using data without polynomial depth calibration. Right: result from same experiment, repeated with polynomial depth image correction. Both maps are generated using pre-computed trajectories that do not rely on depth images

method can only provide acceptable accuracy within 3 m. The error will increase to as high as 6 m in both X and Y axis. For depth Z, the error could increase as high as 2 m, which affect the depth bias prediction a lot.

We repeated the three-room experiment, with and without unwarping the depth images using our calibration model The results of the experiment can be seen in Fig. 21. On the left is the final map result (top-down orthographic projection) with the uncalibrated data. On the right is the result using the unwarped (calibrated) data. The figure demonstrates that the unwarped data produces significantly better results, even with offline graph-based optimization performed in both cases. The systematic skew in the walls of the map in the uncalibrated experiment can be attributed to the slight concavity of the uncalibrated depth data when observing flat surfaces, resulting in incorrect trajectory estimation.

We further explored the benefits of our calibration procedure for systems that use other trajectory estimation methods. In Fig. 22, we present a reconstruction using a dataset obtained with the Peanut device. The trajectory is pre-computed using a combination of visual-inertial odometry and bundle adjustment based solely on the monocular (non-depth) images. Thus, the trajectory is independent of the depth bias. However, removing the bias results in much cleaner scene reconstruction.

7 Conclusions

In this paper, we presented a calibration procedure and uncertainty model for depth readings of RGB-D cameras. The methods we described allow 3D points constructed from depth images to be treated as zero-mean multi-variate Gaussian distributions with a known covariance matrix. This is of interest to any system which performs calculations on RGB-D data where the precision and accuracy are important. We demonstrate experimental evidence of how the calibration procedure affects visual odometry and mapping applications, and demonstrate the predictive power of our uncertainty estimation model, which is able to estimate uncertainties around object edges better than the previously published formulations in this field. The calibration procedure requires that a checkerboard is placed in multiple distances and locations from the camera to obtain dense data, and is reasonable for distances up to 4–5 m.

Further, we presented a visual odometry system for RGB-D cameras. The system uses sparse features which are registered against a persistent model of bounded size. The model is updated through a probabilistic Kalman filter framework. In order to achieve this, we developed a formulation for the 3D uncertainty in sparse features in RGB-D images, based on a Gaussian mixture model of readings in a local image window.

Finally, we presented a place recognition procedure which is used to find correspondences and relative transformations between non-consecutive keyframes for doing pose-graph SLAM. We present two approaches—an exhaustive one and a heuristic one, and evaluate the recognition rate of the latter.

An implementation of our system, developed for use with the ROS (Quigley et al. 2009) framework, is available for download under a free, open-source license from our website (`http://robotics.ccny.cuny.edu/`).

References

Basso, F., Menegatti, E., & Pretto, A. (2018). Robust intrinsic and extrinsic calibration of RGB-D cameras. *IEEE Transactions on Robotics*, *34*(5), 1315–1332.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, *110*(3), 346–359.

Besl, P., & McKay, N. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*, 239–256.

Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*, *25*, 120–125.

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., et al. (2016). The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*. https://doi.org/10.1177/0278364915620033.

Dryanovski, I., Jaramillo, C., & Xiao, J. (2012). Incremental registration of RGB-D images. In *IEEE International conference on robotics and automation (ICRA)*.

Dryanovski, I., Valenti, R. G., & Xiao, J. (2013). Fast visual odometry and mapping from RGB-D data. In *International conference on robotics and automation* (Vol. 10031).

Endres, F., Hess, J., Cremers, D., & Engelhard, N. (2012). An evaluation of the RGB-D SLAM system. *Perception*, *3*(c), 1691–1696.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Garrido-Jurado, S., Munoz Salinas, R. M., Madrid-Cuevas, F., & Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, *47*(6), 2280–2292. https://doi.org/10.1016/j.patcog.2014.01.005.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The kitti vision benchmark suite. In *Conference on computer vision and pattern recognition (CVPR)*.

Google: Project Tango (2014). Retrieved from https://en.wikipedia.org/wiki/Tango_(platform).

Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2010). RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments. In *RGB-D: Advanced reasoning with depth cameras workshop in conjunction with RSS*.

Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2012). RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, *31*(5), 647–663.

Herrera, D., Kannala, J., & Heikkilä, J. (2010). Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(10), 2058–2064. https://doi.org/10.1109/TPAMI.2012.125.

James Bowman, P. M. (2017). ROS camera calibration. Retrieved from http://wiki.ros.org/camera_calibration.

Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., & Aanæs, H. (2014). Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 406–413)

Karan, B. (2015). Calibration of kinect-type RGB-D sensors for robotic applications. *Fme Transactions*, *43*, 47–54.

Kerl, C., Sturm, J., & Cremers, D. (2013). Dense visual slam for RGB-D cameras. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2100–2106). IEEE.

Kerl, C., Sturm, J., & Cremers, D. (2013). Dense visual slam for RGBD-D cameras. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2100–2106). IEEE.

Khoshelham, K., & Elberink, S. O. (2012). Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, *12*(2), 1437–1454. https://doi.org/10.3390/s120201437.

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 2007 6th IEEE and ACM international symposium on mixed and augmented reality* (pp. 1–10). IEEE Computer Society.

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). g2o: A general framework for graph optimization. In *ICRA*. Shanghai.

Meilland, M., & Comport, A. I. (2013). On unifying key-frame and voxel-based dense visual SLAM at large scales. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 3677–3683). IEEE.

Muja, M., & Lowe, D. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *International conference on computer vision theory and application VISSAPP'09* (Vol. 340, pp. 331–340). INSTICC Press.

Mur-Artal, R., & Tardós, J. D. (2017). ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, *33*(5), 1255–1262.

Newcombe, R. A., Davison, A. J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., et al. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *10th IEEE international symposium on mixed and augmented reality (ISMAR)* (pp. 127–136).

Nguyen, C.V., Izadi, S., & Lovell, D. (2012). Modeling Kinect sensor noise for improved 3d reconstruction and tracking. In *Second international conference on 3D imaging, modeling, processing, visualization and transmission (3DIMPVT)* (pp. 524–530). IEEE.

Olesen, S. M., Lyder, S., Kraft, D., Krüger, N., & Jessen, J. B. (2012). Real-time extraction of surface patches with associated uncertainties by means of Kinect cameras. *Journal of Real-Time Image Processing*, *10*, 1–14. https://doi.org/10.1007/s11554-012-0261-x.

Park, J. H., Shin, Y. D., Bae, J. H., & Baeg, M. H. (2012). Spatial uncertainty model for visual features using a Kinect^{TM} sensor. *Sensors*, *12*(7), 8640–8662.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., et al. (2009). ROS: An open-source robot operating system. In *International conference on robotics and automation, Figure 1*.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *IEEE international conference on computer vision (ICCV)* (pp. 2564–2571). https://doi.org/10.1109/ICCV.2011.6126544.

Segal, A., Haehnel, D., & Thrun, S. (2009). Generalized-ICP. In *Robotics: Science and systems*. Seattle, USA.

Shi, J., & Tomasi, C. (1994). Good features to track. In *IEEE computer society conference on computer vision and pattern recognition. Proceedings CVPR '94* (pp. 593–600). https://doi.org/10.1109/CVPR.1994.323794.

Smisek, J., Jancosek, M., & Pajdla, T. (2011). 3D with Kinect. In *IEEE international conference on computer vision workshops (ICCV workshops)* (pp. 1154–1160).

Steinbrucker, F., Sturm, J., & Cremers, D. (2011). Real-time visual odometry from dense RGB-D images. In *IEEE international conference on computer vision workshops (ICCV workshops)* (pp. 719–722). https://doi.org/10.1109/ICCVW.2011.6130321.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems. In *Proceedings of the international conference on intelligent robot systems (IROS).*

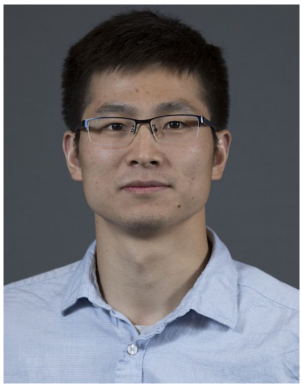
Stutz, D., & Geiger, A. (2018). Learning 3d shape completion from laser scan data with weak supervision. In *IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE Computer Society.

Teichman, A., Miller, S., & Thrun, S. (2013). Unsupervised intrinsic calibration of depth sensors via SLAM. In *Robotics: Science and systems*.

Wang, Y. M., Li, Y., & Zheng, J. B. (2010). A camera calibration technique based on OpenCV. In *3rd International conference on information sciences and interaction sciences (ICIS)* (pp. 403–406).

Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., Davison, A. (2015). Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and systems.*

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



Liang Yang was born in Anqing, China. He received his bachelor degree from Shenyang Aerospace University, Shenyang, China in 2012, and a Ph.D. degree in electrical engineering from the City College of New York (CUNY City College) in 2019, and a Ph.D. degree in pattern recognition and intelligent system from University of Chinese Academy of Sciences in 2019. His research interests cover motion and path planning in complex environment, 3D mapping, visual SLAM, data fusion, and control.

Ivan Dryanovski was born in Sofia, Bulgaria. He received a B.A. in Physics from Franklin and Marhsall College, PA, USA in 2007 and a M.Sc. in Computing Science from Imperial College London, UK in 2009, and Ph.D. in computer science from the Graduate Center of The City University of New York (CUNY), USA in 2015. He currently works at Google Inc. His research interests include computer vision, 3D mapping, SLAM, and quadrotor MAV systems.

Roberto G. Valenti was born in Catania, Italy. He received the M.S. degree in electronics engineering from the University of Catania, Catania, Italy in 2009, and Ph.D. degree in electrical engineering from the City College of New York (CUNY City College) in 2016. He currently works at MathWorks Inc. His research interests include micro-aerial-vehicles (MAV) modeling, simulation, and control.

Dr. George Wolberg is a Professor of Computer Science at the City College of New York. He received his B.S. and M.S. degrees in Electrical Engineering from Cooper Union in 1985, and his Ph.D. degree in Computer Science from Columbia University in 1990. His research interests include image processing, computer graphics, and computer vision. He was an early pioneer of image morphing, and has conducted research on warping, interpolation, registration, 3D reconstruction, and structure from motion. His recent work on PhotoSketch, a photo-centric urban 3D modeling plugin for SketchUp, can be found on www.brainstormllc.com. Prof. Wolberg is the recipient of a 1991 NSF Presidential Young Investigator Award, the 1997 CCNY Outstanding Teaching Award, and the 2000 NYC Mayor's Award for Excellence in Science and Technology. He is the author of Digital Image Warping (IEEE Computer Society Press, 1990), the first comprehensive monograph on image warping and morphing.

Dr. Jizhong Xiao is a Professor of Electrical Engineering at the City College of New York (CCNY/CUNY City College) and a doctoral faculty member of the Ph.D. program in Computer Science at CUNY Graduate Center. He received his Ph.D. degree from Michigan State University in 2002, M.E. degree from Nanyang Technological University, Singapore in 1999, M.S, and B.S. degrees from the East China Institute of Technology, Nanjing, China, in 1993 and 1990, respectively. He started the robotics research program at CCNY in 2002 as the founding director of CCNY Robotics Lab. His current research interests include robotics and control, cyber-physical systems, autonomous navigation and 3D simultaneous localization and mapping (SLAM), real-time and embedded computing, assistive technology, multi-agent systems and swarm robotics. He has published more than 160 research articles in peer reviewed journal and conferences. He received the U.S. National Science Foundation CAREER Award in 2007, the CCNY Outstanding Mentor Award in 2011, and the Humboldt Research Fellowship for Experienced Researchers from the Alexander von Humboldt Foundation, Germany, from 2013 to 2015. He is a senior member of IEEE.