



3D Scene Reconstruction Using Colorimetric and Geometric Constraints on Iterative Closest Point Method

Dong-Won Shin¹ · Yo-Sung Ho¹

Received: 6 October 2016 / Revised: 23 May 2017 / Accepted: 12 July 2017 /

Published online: 7 August 2017

© Springer Science+Business Media, LLC 2017

Abstract Advent of the 3D scene reconstruction framework using the RGB-D camera has enabled users to easily construct their indoor environment in a virtual space and has allowed them to experience an immersive augmented reality with the reconstructed 3D scene. Technically, the early stage of the 3D scene reconstruction framework using the RGB-D camera is based on the frame-to-model registration. It tries to iteratively estimate the transformation parameters of the camera between the incoming depth frame and its previously reconstructed 3D model. However, due to the nature of the frame-to-model registration, the conventional framework has an inherent drift problem caused by the accumulated alignment error. In this paper, we propose a new 3D scene reconstruction framework with the improved camera tracking capability to reduce the drift problem. There are two types of constraints in this work: colorimetric and geometric constraints. For the colorimetric constraint, we impose the more weights on the reliable feature correspondences obtained from color image frames. For the geometric constraint, we compute the consistent surface normal vector for the noisy point cloud data. Experimental results show that the proposed framework reduces the absolute trajectory error representing the amount of the drift and shows a more consistent trajectory in comparison to the conventional framework.

Keywords RGB-D camera · 3D reconstruction · SLAM · augmented reality · iterative closest point method

✉ Yo-Sung Ho
hoyo@gist.ac.kr

Dong-Won Shin
dongwonshin@gist.ac.kr

¹ School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), 123 Cheomdangwagi-ro, Buk-gu, Gwangju 61005, Republic of Korea

1 Introduction

Currently, the augmented reality (AR) system is getting popular due to the releases of its applications in various fields such as gaming, robotics, education, and art. We can place virtual furniture in our room before the actual purchase and at the same time, simulate whether it is compatible to our house or not by mobile AR technology. Moreover, a smart mirror can virtually overlay charming clothes on our body and allow us to conveniently change clothes without any damages on the items.

What is AR exactly? The definition of AR is a live direct or indirect view of a physical, real-world environment whose elements are augmented by computer-generated sensory inputs such as sound, video, graphics or GPS data [11]. The statement implies we can overlay the augmented information over the real scene and interact with them as those exist right in front of us. In order to compose the augmented information on the real scene seamlessly, we can consider the 3D scene reconstruction as a core of the vision-based AR techniques.

An RGB-D camera captures the information of the scene including the complementary nature of the depth and color images. The release of this renowned device incredibly achieved the advancement in various research fields such as the object tracking and recognition, human activity analysis, hand gesture recognition, and indoor 3-D mapping [14]. Especially for the indoor 3-D mapping, also known as the 3D scene reconstruction, RGB-D camera has played the important role in transferring the real scene to the virtual space at a high frame rate.

In the modern 3D scene reconstruction framework using RGB-D camera, it is important to estimate an accurate camera trajectory since understanding where the camera exists in the 3D scene directly affects the quality of the reconstructed 3D scene [7]. Many of them rely on 3D point cloud registration on a frame-to-model basis in an early phase for an initialization of camera positions [16, 19, 27]. However, since there is a model drift problem due to an accumulated trajectory error derived from the frame-to-model approach, this error should be reduced for the consistent quality of 3D model.

In this paper, we focus on the 3D scene reconstruction framework using an RGB-D camera to generate a more consistent 3D model of the scene. Our main contribution consists of the two-fold of colorimetric and geometric constraints. Briefly, the colorimetric constraint is to give more weight on the feature correspondences from a color frame and the geometric constraint is to estimate the robust surface normal vector for the noisy point cloud data. We will introduce details of our framework in the following sections.

2 Related Works

In order to reconstruct the 3D scene in the virtual space, various approaches have been proposed for several decades. The early method for 3D scene reconstruction using RGB-D camera is KinectFusion [23]. It incrementally registers the incoming depth information from the hand-held RGB-D camera and reconstructs the 3D model as a real time by the benefit of GPU parallel processing. Even it was an early strategy of the 3D scene reconstruction, it has become the main branch of the field.

A flowchart of the 3D scene reconstruction system is shown in Fig. 1. After capturing the 2-dimensional depth frames from the RGB-D camera, we can generate the 3-dimensional point cloud data (PCD) with the camera's intrinsic parameters. Then we estimate normal vectors for each point by the simple cross product calculation with the neighboring vertices. By using the PCD and normal vectors, we can perform the frame to frame registration and compute the camera trajectories via the point-to-plane iterative closest point (ICP) method [20]. After that, we compute the volumetric

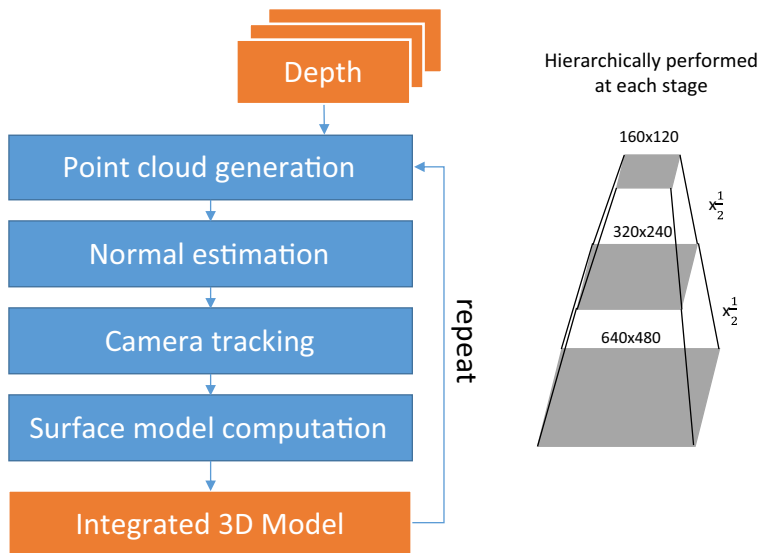


Fig. 1 Flowchart of the conventional 3D scene reconstruction

model of the scene and finally, we can obtain the integrated 3D model [6]. The overall procedure is repeated as depth frames are coming and hierarchically performed at each stage. Thanks to the benefits of the GPU parallel programming, it can be operated in real-time.

However, there is a drift problem caused by accumulated pose estimation errors in the conventional framework. It requires a costly offline global optimization step and makes users generate explicit loop closures in the trajectory to allow gross alignment errors to be solved. Moreover, the conventional method only uses the depth information from the RGB-D camera even if the synchronized and view-aligned color information is available. Lastly, when the surface normal vectors are estimated, the conventional method uses the simple cross product calculation which is vulnerable to noisy PCD. Those problems of the conventional method cause a globally inconsistent 3D model.

Henry et al. proposed the RGB-D Mapping method combining both the color and depth information to track the camera position in 3D space [16]. They found the initial transformation between frames with arbitrary visual features and optimized the color and depth combined cost function iteratively. However, they didn't provide the comprehensive analysis for the weight value.

In order to reduce the drift problem, Fioraio et al. introduced the sub-volume based registration method [8]. The sub-volumes are generated every K frames and their poses are globally optimized through a volume blending scheme. However, the consistent frame to frame registration is still necessary for the sub-section, which we are going to experiment with the various size of sub-volumes.

Especially for the ICP procedure, which is what we are concentrating on, the point-to-plane method is widely used and it is included in the general 3D scene reconstruction framework [23]. Aside from that, Guy Godin et al. proposed an ICP method imposing the different weight depending on the Euclidean distance between the points [10]. The inner product of corresponding normal vectors can be used for ICP weighting scheme [12]. We will compare the camera trajectory result with these reference methods in Section 4.

In this paper, we present a 3D scene reconstruction algorithm with the colorimetric and geometric constraints on the iterative closest point method to reduce the inherent drift problem. The rest of the paper is organized as follows. In Section 3, we describe the core of our 3D scene reconstruction

algorithm in detail. In Section 4, we show the experiment setups and results of the proposed method. Finally, in Section 5 we conclude and discuss the limitations of our method and future directions.

3 Proposed 3D Scene Reconstruction System

In this section, we introduce the proposed 3D scene reconstruction system in detail. Fig. 2 shows the flowchart of the proposed 3D scene reconstruction framework. Mainly, the two constraints are newly included.

In the case of the colorimetric constraint, the main purpose of this constraint is to impose more weight on the reliable colorimetric feature correspondences during the camera tracking step. We first extract features by the SURF detector from current and previous color frames to take the benefit of the algorithmic acceleration [1]. The correspondence matching step is performed by a fast nearest neighbor method with the feature descriptors [21], and outlier removal step is conducted by RANSAC technique with the homography constraint [9]. The refined correspondences will be used for the camera tracking step, especially in our weighted point-to-plane ICP method.

In the case of the geometric constraint, the primary goal is to estimate the consistent normal vectors on the noisy PCD. Therefore, we perform the normal estimation method using the principle component analysis (PCA) with neighboring vertices instead of using the simple cross product in the conventional framework. At this point, the size of neighbor is critical since it determines the amount of noise and curvature from other primitives.

In order to obtain the appropriate size of a neighbor set, we exploit the distance transform map on the edge image. We first obtain the edge information from both color and depth images by Canny edge detector [5]. The distance information from a specific pixel to the nearest edge is calculated by distance transform on the binary edge image [4]. Finally, the size of the neighbor set at each pixel is defined by the distance information.

3.1 Colorimetric Constraint

Prior to the explanation of the proposed colorimetric constraint, we will briefly introduce point-to-plane iterative closest point method that our method is originated from and then discuss the weighted point-to-plane iterative closest point method.

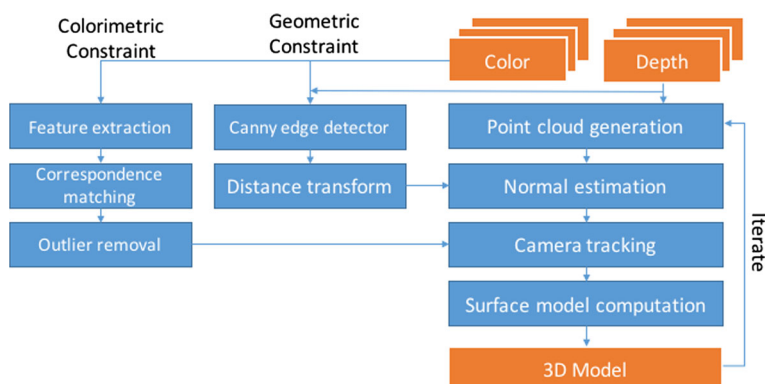


Fig. 2 Flowchart of the proposed 3D scene reconstruction framework

3.1.1 Conventional Point-to-Plane Iterative Closest Point Method

The Iterative Closest Point (ICP) method is to find the optimum transformation relationship containing rotation and translation between two sequential PCD [3]. In the conventional framework, the Point-to-Plane ICP using the perpendicular distance from the source point to the tangent plane at the destination point was exploited. Fig. 3 visualize the Point-to-Plane ICP and (1) illustrates the cost function for the relation.

$$\mathbf{M}_{\text{opt}} = \underset{\mathbf{M}}{\operatorname{argmin}} \sum_i ((\mathbf{M} \cdot \mathbf{s}_i - \mathbf{d}_i) \cdot \mathbf{n}_i)^2 \quad (1)$$

What we want to find is the optimum transformation matrix \mathbf{M} containing the rotation matrix and translation vector between the source PCD $\mathbf{s}_i = \{s_{ix}, s_{iy}, s_{iz}\}$, and destination PCD $\mathbf{d}_i = \{d_{ix}, d_{iy}, d_{iz}\}$, which is minimizing the perpendicular distance with normal vector $\mathbf{n}_i = \{n_{ix}, n_{iy}, n_{iz}\}$.

In order to find the optimum transformation, K. L. Low solved the cost function by the linear approximation of the rotation matrix and the least-square method [20]. Equation (2) shows the modified form of (1) in a least-square sense.

$$\mathbf{x}_{\text{opt}} = \underset{\mathbf{x}}{\operatorname{argmin}} |\mathbf{A}\mathbf{x} - \mathbf{b}|^2, \quad (2)$$

$$\text{where } \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & n_{1x} & n_{1y} & n_{1z} \\ a_{21} & a_{22} & a_{23} & n_{2x} & n_{2y} & n_{2z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & n_{Nx} & n_{Ny} & n_{Nz} \end{pmatrix} \quad \text{with } \begin{aligned} a_{i1} &= n_{iz}s_{iy} - n_{iy}s_{iz} \\ a_{i2} &= n_{ix}s_{iz} - n_{iz}s_{ix} \\ a_{i3} &= n_{iy}s_{ix} - n_{ix}s_{iy} \end{aligned}$$

$$\mathbf{b} = \begin{pmatrix} n_{1x}d_{1x} + n_{1y}d_{1y} + n_{1z}d_{1z} - n_{1x}s_{1x} - n_{1y}s_{1y} - n_{1z}s_{1z} \\ n_{2x}d_{2x} + n_{2y}d_{2y} + n_{2z}d_{2z} - n_{2x}s_{2x} - n_{2y}s_{2y} - n_{2z}s_{2z} \\ \vdots \\ n_{Nx}d_{Nx} + n_{Ny}d_{Ny} + n_{Nz}d_{Nz} - n_{Nx}s_{Nx} - n_{Ny}s_{Ny} - n_{Nz}s_{Nz} \end{pmatrix},$$

$$\mathbf{x} = (\alpha \quad \beta \quad \gamma \quad t_x \quad t_y \quad t_z)^T.$$

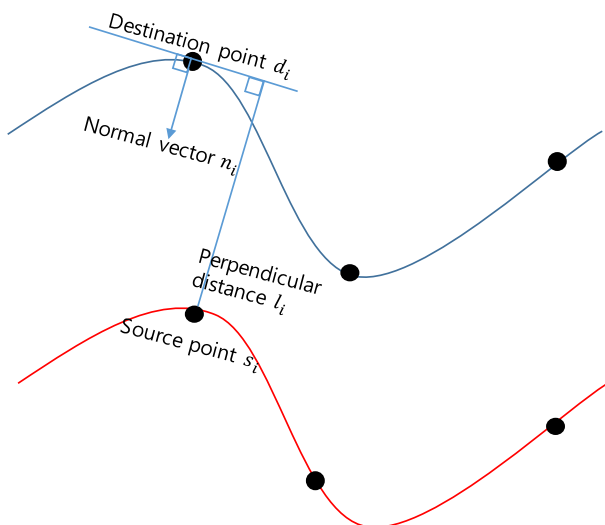


Fig. 3 Visualization of Point-to-Plane ICP

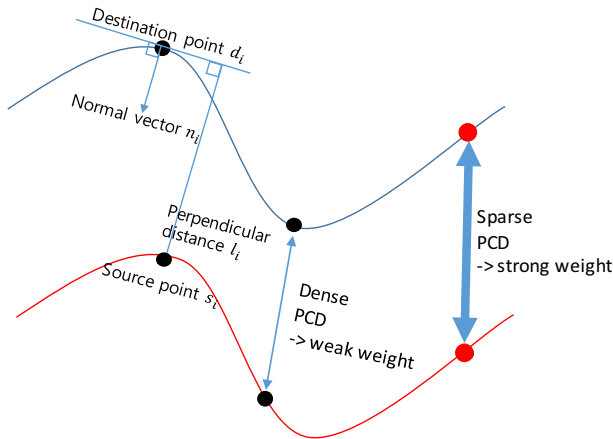


Fig. 4 Visualization of weighted Point-to-Plane ICP

In this case what we want to find is the optimum solution for \mathbf{x} including the rotation $(\alpha \ \beta \ \gamma)$ and translation $(t_x \ t_y \ t_z)$ parameters. It can be simply solved by the singular value decomposition (SVD) and the optimum will be found by

$$\mathbf{x}^* = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}. \quad (3)$$

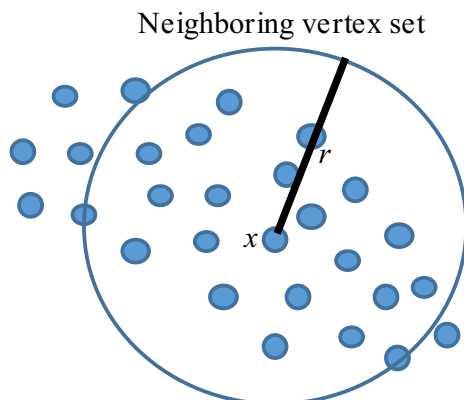
3.1.2 Proposed Weighted Point-to-Plane Iterative Closest Point Method

In the proposed framework, we exploited the weighted Point-to-Plane ICP to impose more weight value on feature correspondences. Equation (2) can be reproduced in a weighted least-square fashion like (4).

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^N \|\mathbf{A}_i \mathbf{x} - \mathbf{b}_i\|^2 \quad (4)$$

where w_i represents the weight on the i -th correspondence. In the proposed method, the weight values are different depending on the type of the PCD. Specifically, there are two types of PCD: dense and sparse PCD. The dense PCD represents the point cloud created from the depth frames and

Fig. 5 Visualization of the neighboring vertex set



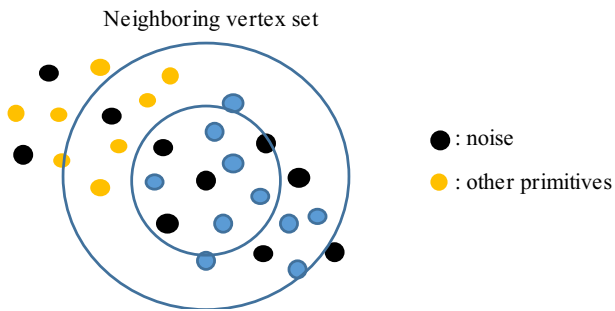
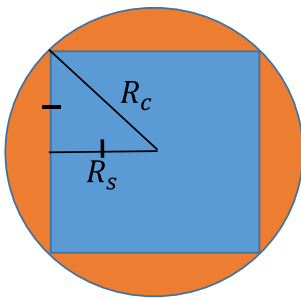


Fig. 6 Large and small neighboring vertex set



R_c : a radius of a circumcircle

R_s : (a side of a square)/2

$$R_s^2 + R_s^2 = R_c^2$$

$$\therefore R_s = \frac{R_c}{\sqrt{2}}$$

Fig. 7 Relationship between a radius of a circumcircle and a side of a square

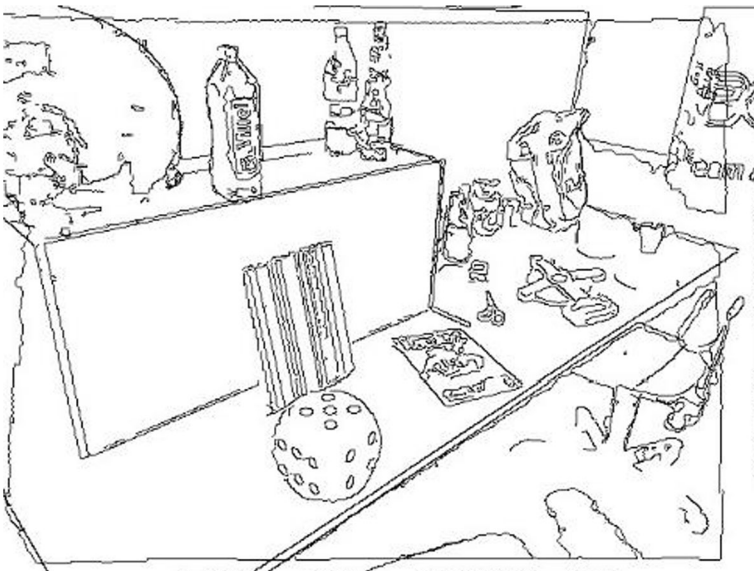


Fig. 8 Example of Canny edge detector

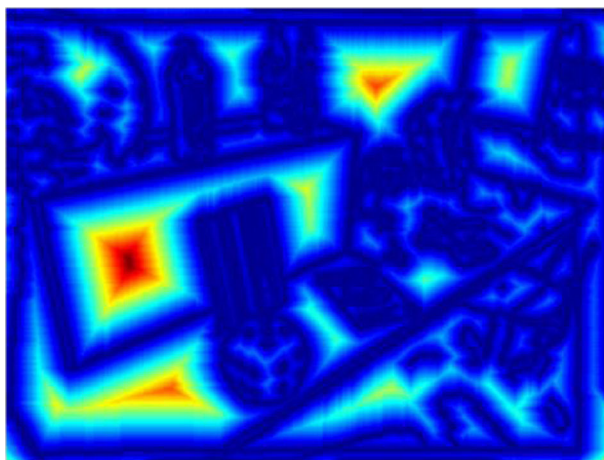


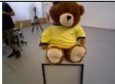





Fig. 9 Example of the distance transform on the edge

the sparse PCD represents the point cloud from the features in the color frames. Fig. 4 illustrates the weighted Point-to-Plane ICP.

We can represent the cost function by the residual matrix.

$$E_w(x) = w_i \|\mathbf{A}_i \mathbf{x} - \mathbf{b}_i\|^2 = \sum_{i=1}^N w_i r_i^2 = \mathbf{r}^T \mathbf{W} \mathbf{r} = \|\mathbf{W}^{1/2} \mathbf{r}\|^2 \quad (5)$$

Table 1 Datasets

ID	Name	Thumbnail	Total length (frames)	Avg. Speed (m/s)	Image size	Camera Parameters*
1	freiburg3_teddy		2344	0.248	640x480	fx: 525.0 fy: 525.0 cx: 320.0 cy: 320.0
2	freiburg3_long_office		2509	0.249	640x480	
3	freiburg2_xyz		3666	0.058	640x480	
4	freiburg1_desk1		595	0.413	640x480	
5	kt0_office_room		1510	0.126	640x480	fx: 481.2 fy: -480.0
6	kt0_living_room		1510	0.126	640x480	cx: 319.5 cy: 239.5

*(fx, fy) : focal length, (cx, cy): principal point

By using the vector derivative,

$$\frac{\partial E_w(\mathbf{x})}{\partial \mathbf{x}} = (\mathbf{b} - \mathbf{A}\mathbf{x})^T (\mathbf{W} + \mathbf{W}^T) (-\mathbf{A}) = -2(\mathbf{b} - \mathbf{A}\mathbf{x})^T \mathbf{W} \mathbf{A} = 0 \quad (6)$$

Finally, we can find the optimum value x^* for the weighted least-square method by finding the location making the cost function to zero.

$$\mathbf{x}^* = (\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W} \mathbf{b} \quad (7)$$

3.2 Geometric Constraint

The surface normal vector is an extremely useful input for reconstruction methods and the incorrect normal vectors will result in an erroneous 3D model. Therefore, it is important to obtain the consistent surface normal vector even in the situation of noisy PCD which is common in the modern depth cameras (structure light sensor, time-of-flight camera, Lidar sensor) [2].

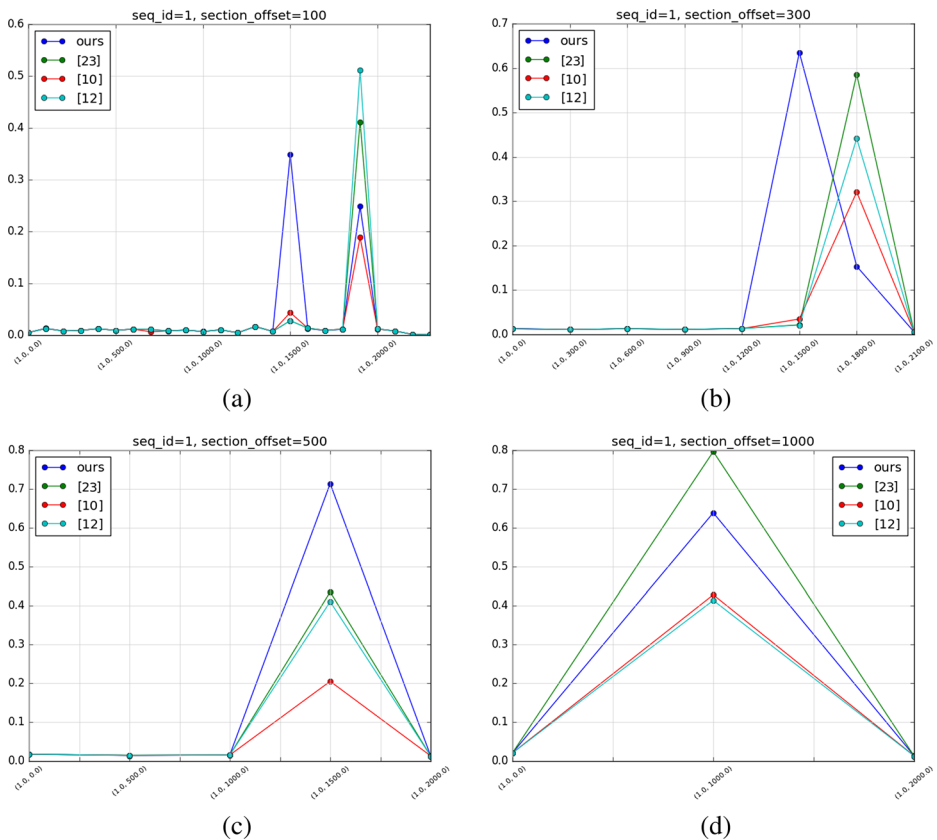


Fig. 10 ATE comparison graph for the sequence ID 1: even the ATE results look similar, but the proposed method is better with the slightly reduced error when those show close results

3.2.1 Normal Estimation Using Principle Component Analysis

The conventional framework, however, includes the simple cross product calculation vulnerable to the noisy PCD. Hence the robust normal estimation method is necessary. In this paper, we exploit the normal estimation method using the principle component analysis (PCA) to elaborate the noisy normal vectors [24].

First, we need to compute the centroid \bar{x} for the neighboring vertex set $\{x_1, x_2, \dots, x_k\}$ within the radius r of the query point x . Then, a covariance matrix C is obtained by

$$C = \frac{1}{k} \sum_{i=1}^k (\bar{x} - x_i)(\bar{x} - x_i)^T \quad (8)$$

Fig. 5 visualizes the neighboring vertex set with the notations. After applying SVD on the covariance matrix, the normal vector for the query point is defined by the eigenvector corresponding to the smallest eigenvalue. Due to the nature of the PCA, the orientations of normal vectors are not consistent for the scene and it should be corrected by the viewpoint of the camera v_p .

$$n \cdot (v_p - x) > 0 \quad (9)$$

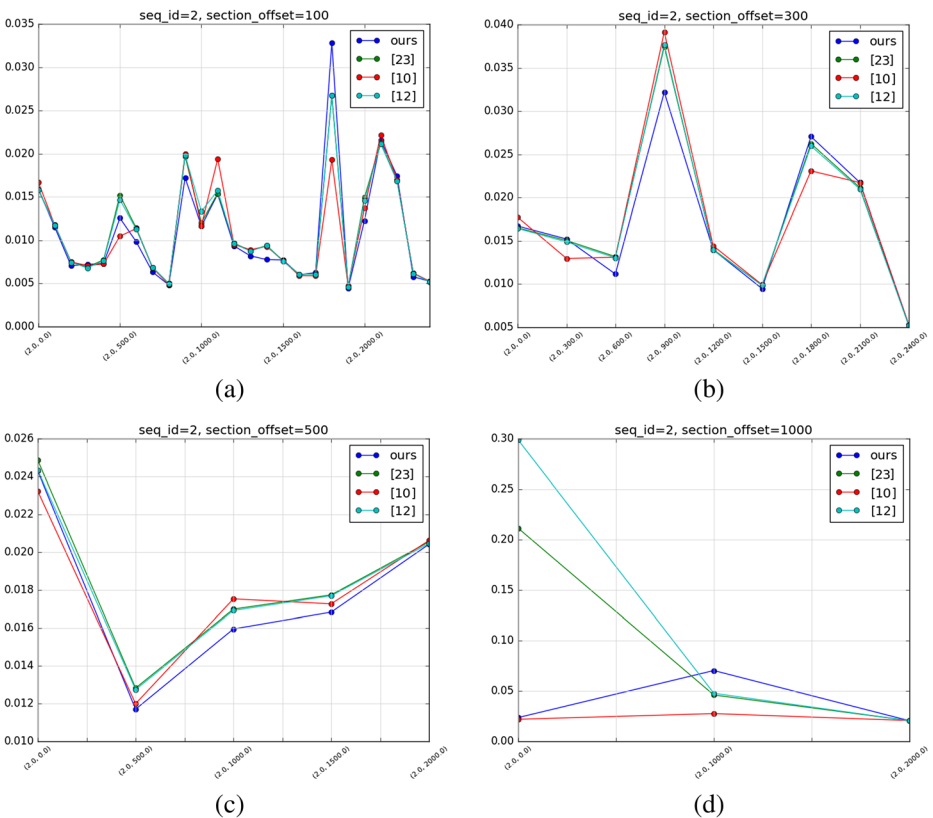


Fig. 11 ATE comparison graph for the sequence ID 2: The proposed method performs better or equal results compared to the other methods

If the estimated normal \mathbf{n} do not satisfy (9), it should be flipped by multiplying -1 . Then we can finally get the consistent normal vectors for the point cloud. Aside from that, Hugues Hoppe et al. employed minimum spanning tree to get the consistent tangent plane orientation [18]. However, it is accurate but very time-consuming method and not appropriate to the real-time application.

3.2.2 Neighborhood Selection

The critical issue on this approach is the size of the neighboring vertex set. Since we consider the depth frames rather than unorganized 3D vertices, the size can be defined by the radius r in the 2D image coordinate. In this case, if the size is too small, the noise can dominate the reconstruction result, otherwise, the points from other primitives having different depth values can be included, as shown in Fig. 6. Therefore, the appropriate size of the neighboring vertex set is required and there are several approaches to determine the size. C. Weber et al. insisted that a neighborhood with a size of radius 16 performs best [26]. S. Gumhold et al. proposed radius 10 to 16 and J. Wu et al. exploited radius 12 [13, 28].

In this paper, however, we will use an adaptive size of the radius rather than the fixed size. S. Holzer et al. proposed the adaptive neighborhood selection for the surface normal estimation

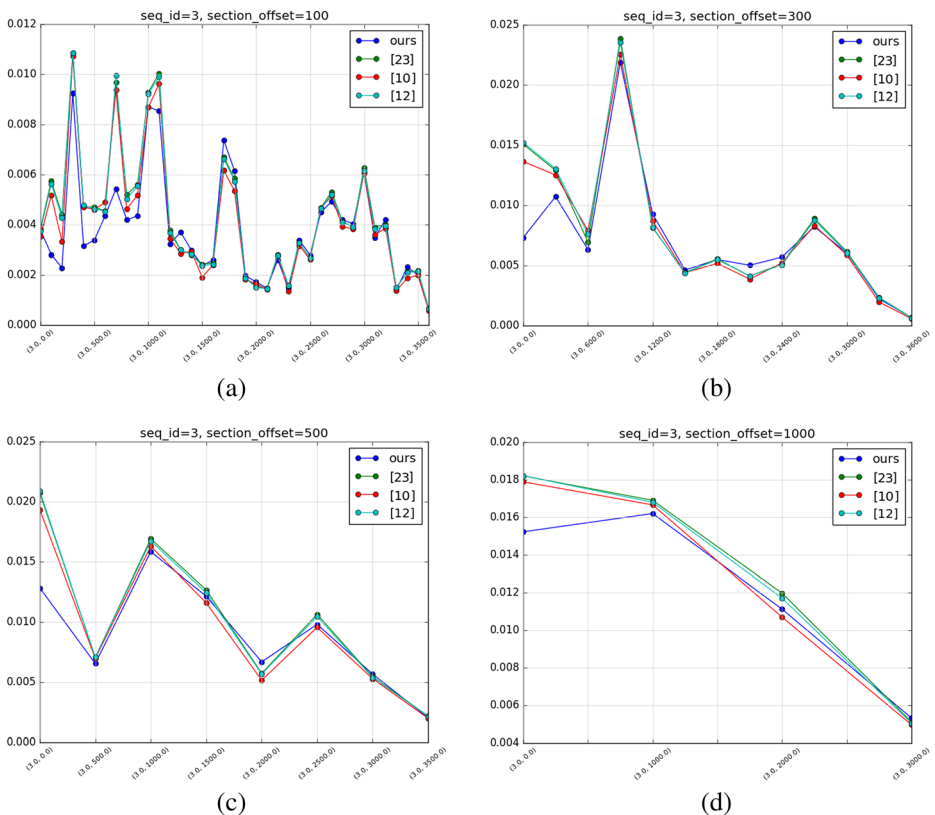


Fig. 12 ATE comparison graph for the sequence ID 3: This sequence has the longest trajectory and slowest speed. In the slow speed scenario, the proposed method is slightly better than the conventional methods even though those already show the good ATE result

[17] and the main contribution of their research to the adaptive size of the neighborhood is two-fold: depth-dependent smoothing area map and depth change indication map.

The depth-dependent smoothing area map implies the neighborhood size should be dependent on the depth value at each pixel since the resolution of the depth gets worse as increasing depth. Equation (10) represents the response of the depth-dependent smoothing area map $B(m,n)$:

$$B(m,n) = \alpha \cdot \beta \cdot D(m,n)^2 \quad (10)$$

where α can be defined by the sensor characteristics and β is a user controllable value. $D(m,n)$ is the depth value at (m,n) position in the image coordinate.

Next, the depth change indication map suggests the neighborhood size should be dependent on the distance from the distinguishable depth change. It is simply constructed by the depth change detection threshold. The depth change indication map $C(m,n)$ can be constructed as

$$C(m,n) = \begin{cases} 1 & \left\{ \begin{array}{l} \text{if } \|\delta D_x(m,n)\| \geq \text{threshold} \\ \text{or } \|\delta D_y(m,n)\| \geq \text{threshold} \end{array} \right. \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

If the depth change at (m,n) position exceeds some threshold value, one is recorded on the position, otherwise it will be zero.

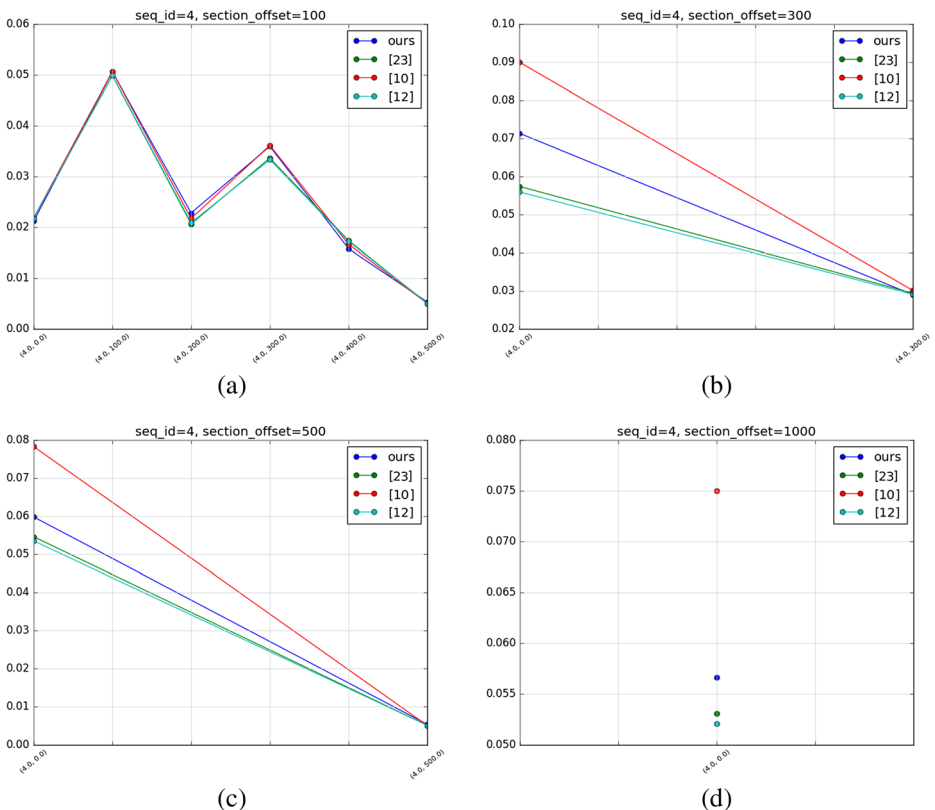


Fig. 13 ATE comparison graph for the sequence ID 4: This sequence has the shortest trajectory and fastest speed

The final smoothing area map can be defined by the combination of the depth smoothing area map and depth change indication map. Equation (12) represents the final neighboring area map $R(m,n)$:

$$R(m,n) = \min\left(B(m,n), \frac{T(m,n)}{\sqrt{2}}\right) \quad (12)$$

where $T(m,n)$ is the distance transform map on depth change indication map $C(m,n)$ [4]. The root of two ($\sqrt{2}$) is the dividing factor for the rectangular radius shown in Fig. 7. The details of the adaptive size of the neighborhood is explained in [17].

Inspired by their research, we determine the size of neighbor in an adaptive neighborhood selection manner. Unlike their approach, we exploit Canny edge detector to get the depth change on the depth frame. In addition to that, the intensity change on the color frame is exploited since the minor depth changes are hard to be detected on the depth frame only. After that, we obtained the distance transform map $T'(m,n)$ on the edge image. Lastly, we exploited the maximum normal radius instead of the depth-dependent smoothing area map since it is mostly controlled by the user input. The modified neighboring area map $R'(m,n)$ is described in (13).

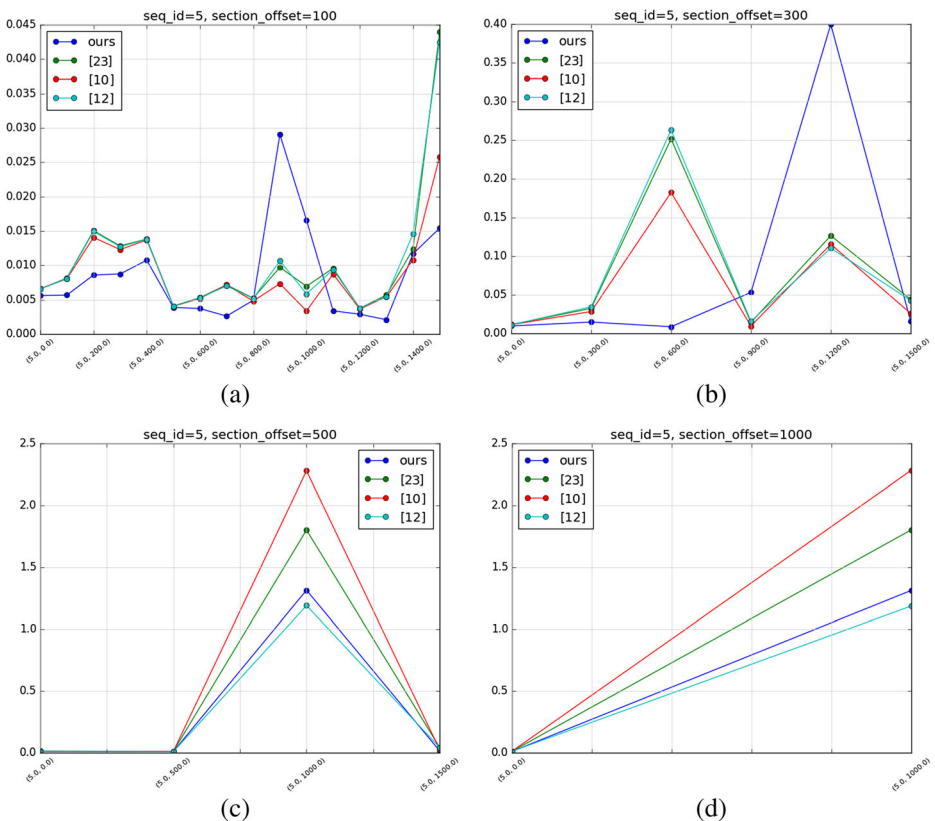


Fig. 14 ATE comparison graph for the sequence ID 5: The sequence ID 5 and 6 are computer generated 3D models including Gaussian noise. Even though it is in a noisy case, the proposed framework performs better than the conventional ones

$$R'(m, n) = \min \left(\text{maximum_radius}, \frac{T'(m, n)}{\sqrt{2}} \right) \quad (13)$$

Fig. 8 shows the example of Canny edge detector and Fig. 9 shows the example of distance transform on the edge image.

3.2.3 Discarding of the Depth near the Boundary

The depth information near the object boundary is unstable and flickering due to the characteristic of a depth sensor and it will directly affect the quality of the reconstructed 3D scene. Therefore, we would discard the depth information near the boundary when we estimate the camera trajectory. Since we already calculated the distance transform map in Section 3.2.2, we can easily discard the depth information having a $T'(m, n)$ value less than a threshold. In our experiment, we exploited the threshold value as 5.

4 Experiment

We verify the robustness of our proposed framework by comparing it to other various methods in the following sections. In Section 4.1, we introduce a methodology we followed to evaluate

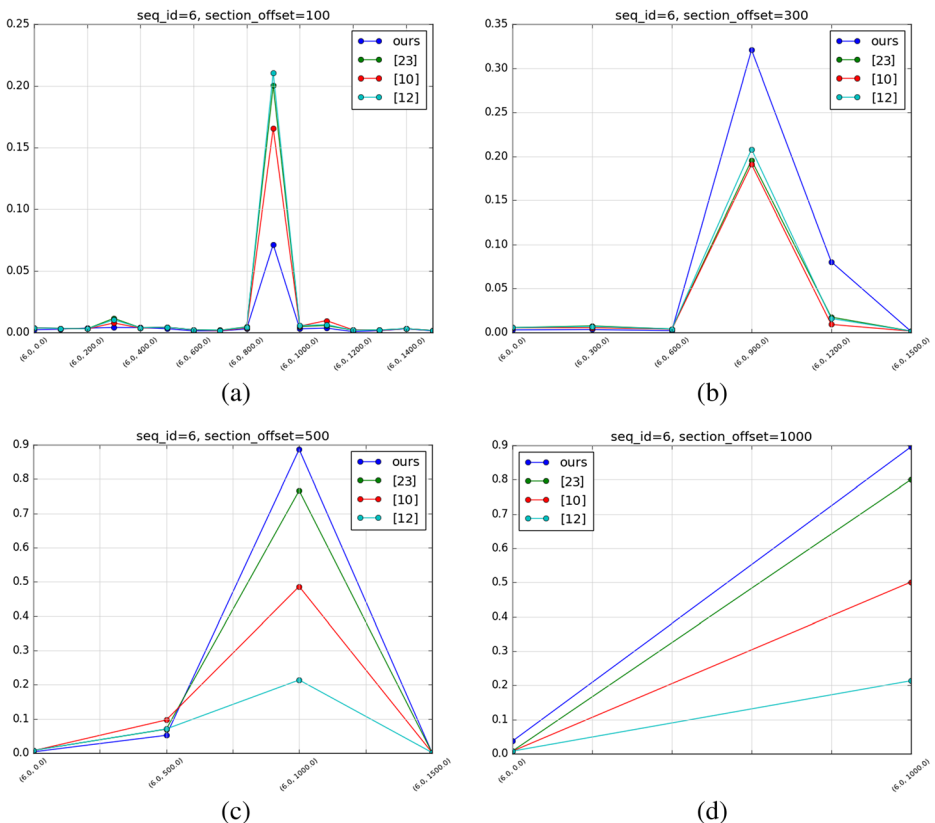


Fig. 15 ATE comparison graph for the sequence ID 6

our framework. Next, we demonstrate the numerical analysis in Section 4.2 and a visual comparison is shown in Section 4.3.

4.1 Methodology

Even though we are focusing on the generic indoor 3D scene reconstruction framework, there are many details in the experiment. Here, we describe the methodology we followed to evaluate our framework.

- 1) **Datasets:** we tested our system with six publicly available datasets as shown in Table 1. We exploited the RGB-D datasets from two sources: Technical University of Munich (TUM) [25] and Imperial College London (ICL) [15]. The sequences with ID 1 to 4 are from TUM which they provide the RGB-D images with ground-truth camera trajectories. The sequences with ID 5 and 6 are from ICL. Unlike the TUM datasets, those are RGB-D data from computer generated 3D scene model with Gaussian noise in their depth images. In this paper, we chose the sequences with the noise in PCD to show the robustness.
- 2) **Ground truth:** All of the introduced datasets have the ground-truth camera trajectory with timestamps at each RGB-D frame. Especially for the TUM datasets, the external motion capture system is used to acquire the pose of the RGB-D camera in 3D space [25]. Special passive markers are attached on the RGB-D camera and the motion capture system tracks the position of the markers by triangulation.

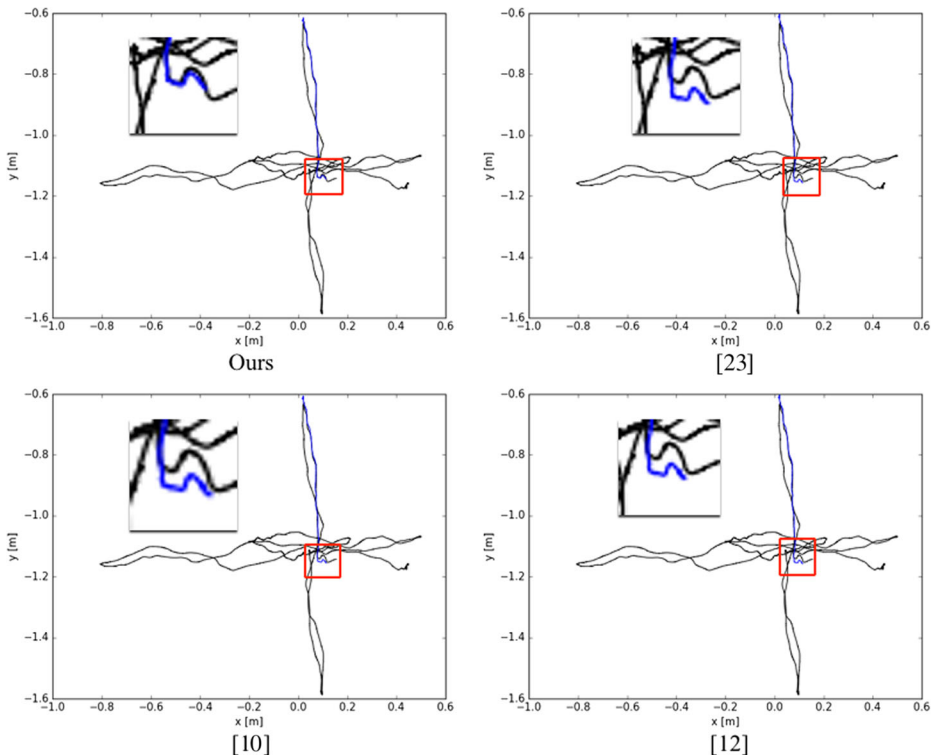


Fig. 16 The estimated trajectory for the sequence ID 3 with a section 0–300: the proposed method shows more consistent result especially on the beginning of the trajectory

- 3) Evaluation measure: We evaluate the correctness of the proposed framework by the Absolute Trajectory Error (ATE) metrics [25]. We first align the estimated and ground-truth trajectories with the timestamps and then evaluate the absolute pose differences between them. The Root Mean Squared Error (RMSE) over all timestamps of the translational components will be compared.
- 4) Control parameters: There are two types of the constraint as we mentioned: the colorimetric and geometric constraints. For the colorimetric constraint, we can control the weights for dense and sparse PCD. The weight value for dense PCD is 0.2 and the weight value for sparse PCD is 0.8. For the geometric constraint, the maximum radius for a neighboring pixel is set to 10 and the threshold value for the discarding unreliable depth information is set to 5.
- 5) Various length of the subsection for subvolumes: Considering the sub-volume based registration method that we discussed it in Section 2, we had the experiments on the various size of the subsection for subvolumes $\{100, 300, 500, 1000\}$. That is, we divided the whole frame by the specific size of subsections, and then compared ATE among various methods on each subsection.
- 6) System settings: The experiments are conducted on a PC with a CPU of Intel core i7–4960 3.60GHz, and is accelerated by a GPU of NVidia Geforce GTX 1070.
- 7) Implementation settings: We implemented the proposed framework based on the benchmark implementation of 3D scene reconstruction, also known as Slambench [22]. It is the publicly-available software framework of 3D scene reconstruction including the well-

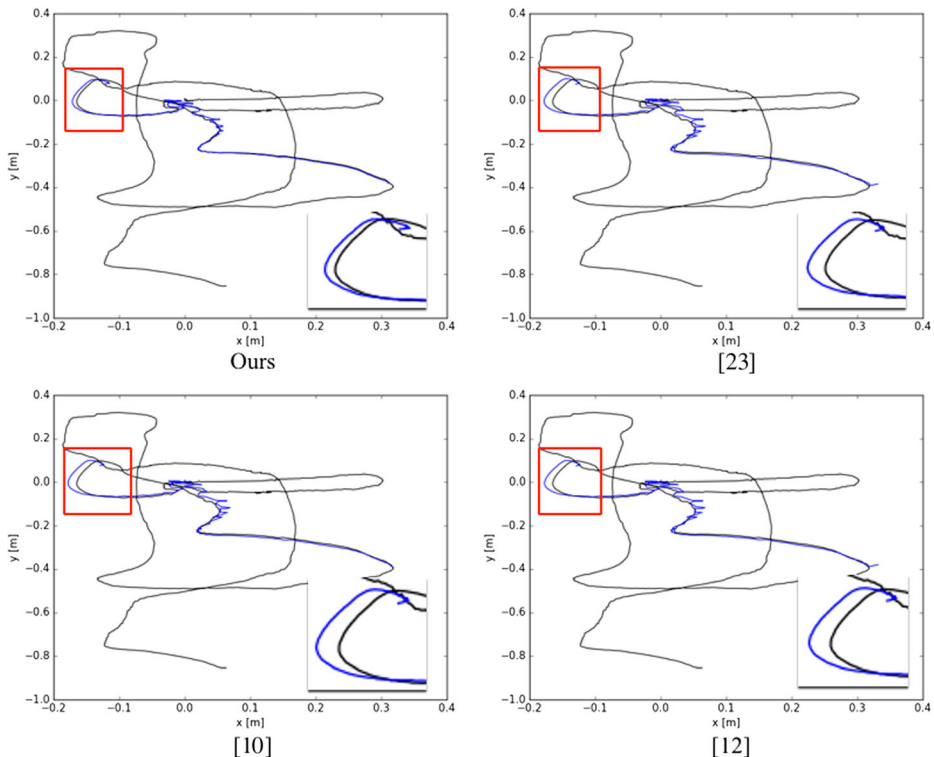


Fig. 17 The estimated trajectory for the sequence ID 5 with a section 300–600: on the curve of the trajectory, the proposed method show a more closer result to the ground-truth trajectory

organized benchmark module. We modified the input module to process the designated RGB-D datasets and extended the core 3D scene reconstruction module as we proposed.

4.2 Comparison Graph of Absolute Trajectory Error

In this section, we describe the ATE comparison graph among several methods for the various sizes of subsection from Figs. 10, 11, 12, 13, 14, and 15. In those figures, the horizontal axis represents the subsections and the vertical axis means the ATE value. Although the results from the conventional methods look close to results from proposed method, the proposed method is better among the comparisons showing the small difference. The detailed numerical ATE values are shown in Appendix. In the appendix, the bold character represents the minimum ATE value.

4.3 Visual Trajectory Comparison

We show the visualization of the estimated trajectory results representing the explicit difference from Figs. 16, 17, 18, and 19. The blue curve represents the estimated trajectory from each method and the black curve represents the ground-truth trajectory. As we can see in these figures, the proposed method shows a more consistent trajectory than the conventional ones.

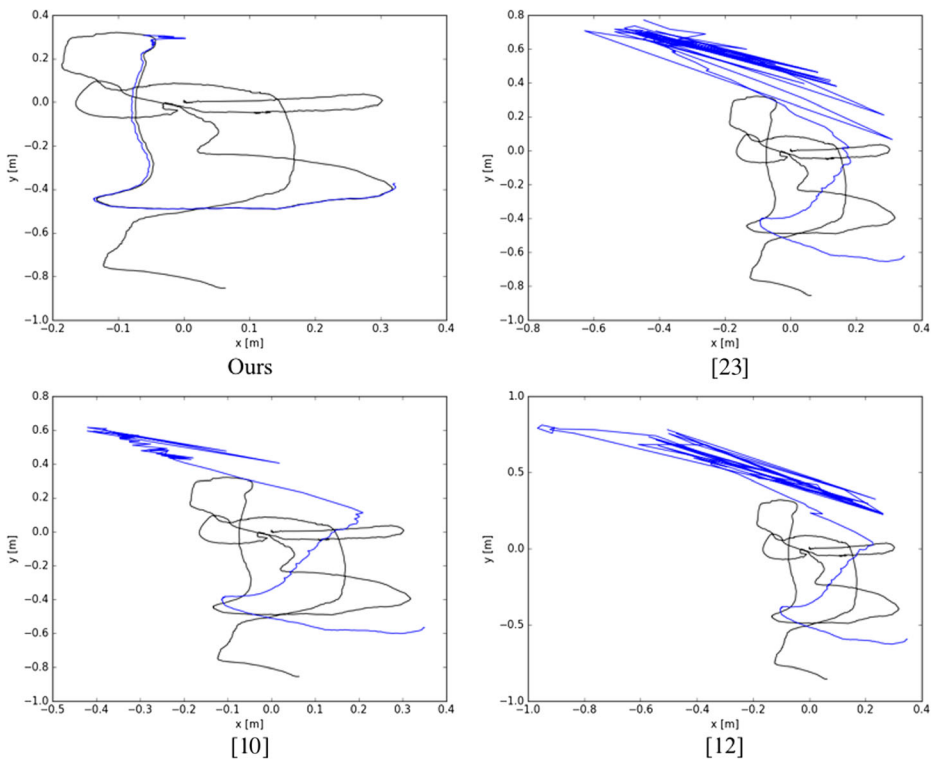


Fig. 18 The estimated trajectory for the sequence ID 5 with a section 600–900: only the proposed method shows the consistent result rather than conventional methods

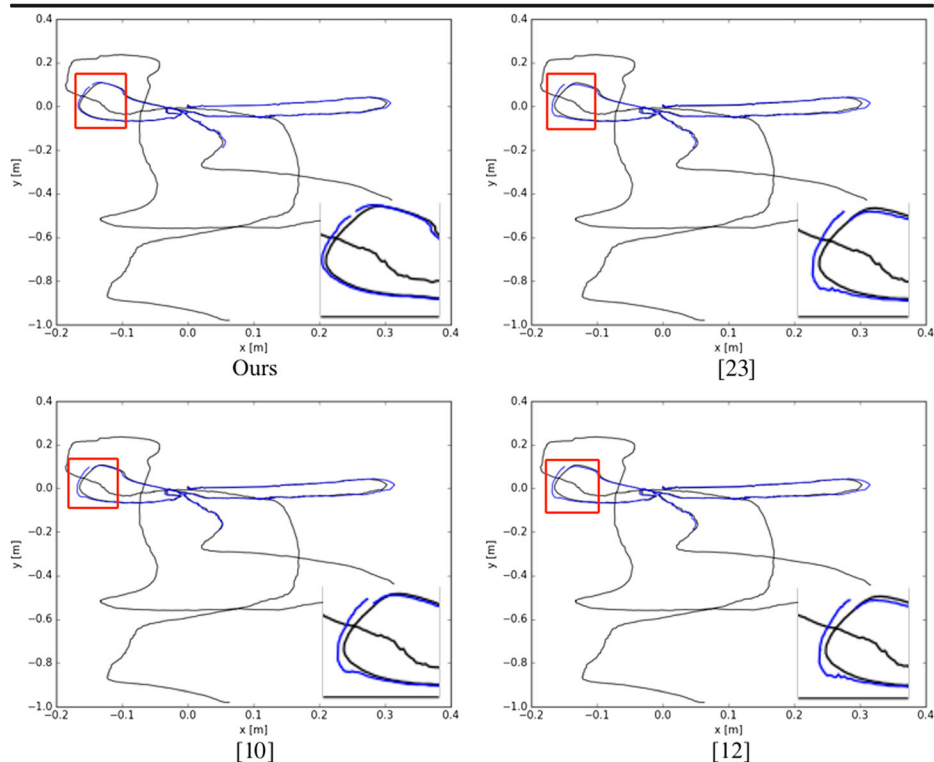


Fig. 19 The estimated trajectory for the sequence ID 6 with section 0–500

4.4 Reconstructed 3D Scene

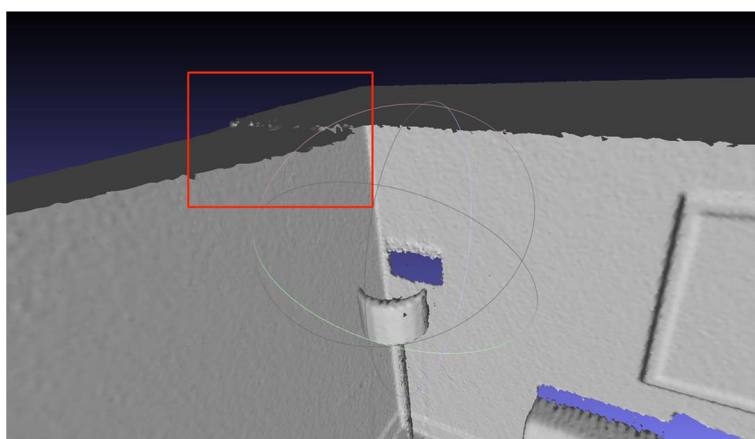
In this Section, we show the reconstructed 3D scene models to visually compare the differences. Figure 20 shows the reconstructed 3D scene models for the sequence ID 6 with section 300–600. As you can see in this figure, the crack on the ceiling exists in the result of [23], but not in the proposed method.

4.5 Elapsed Time

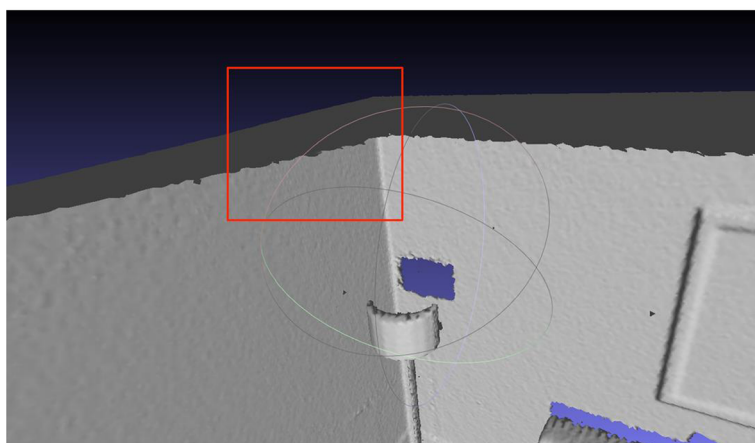
In order to evaluate a time complexity, we displayed the total elapsed time in Table 2. At this point, the preprocessing time includes the time for the feature extraction, photometric correspondence matching, outlier removal, edge detection and distance transform, which are mainly operated on CPU. The tracking time includes the core ICP process such as the vertex correspondence matching, iterative optimization, which are mainly operated on GPU device in parallel. The unit for time is a millisecond. As you can find in this table, the elapsed time for the overall process looks promising thanks to the parallel processing.

4.6 Discussion

Currently, the augmented reality field is getting popular and aggressively extends its area to many applications such as the education, game, industry, and commerce. The augmented reality is such



(a) [23]



(b) ours

Fig. 20 The reconstructed 3D scene models for the sequence ID 6 with section 300–600

a broad research topic including the 3D scene reconstruction, camera tracking, human-computer interaction, computer graphics rendering and so on. Among them, the acquisition of the accurately reconstructed 3D scene is the starting base of the augmented reality since virtual interactive characters can be seamlessly composed after understanding the geometric 3D scene. If there is a crack like Fig. 20(a) in the reconstructed 3d scene, this small difference would invoke an inconsistent feeling from the augmented reality contents.

Table 2 Elapsed time for each sequence (millisecond)

	seq_id = 1	seq_id = 2	seq_id = 3	seq_id = 4	seq_id = 5	seq_id = 6	mean
preprocessing	63.64	78.07	76.29	77.13	53.44	55.93	67.42
tracking	87.77	105.54	98.19	104.71	104.72	105.97	101.15

5 Conclusion

In this paper, we have presented the extending ideas against the conventional 3D scene reconstruction method. The conventional method only exploited the depth information from the RGB-D camera to reconstruct the 3D scene. However, the proposed method not only use the color information from the camera but also refine the normal vector of the 3D vertices by PCA algorithm and the distance information to reconstruct the more consistent 3D model of the scene. The qualitative experiments demonstrate that our method shows more consistent results in terms of the camera trajectory which is considered as important as the dual problem of the 3D reconstruction problem. The users can generate the more consistent model of the 3D scene by using the proposed method and better experience the immersive feeling of the augmented reality system.

Acknowledgements This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) (No. 2011-0030079)

Appendix

Comparison table of the absolute trajectory error

Sequence ID 1:

Section_offset=100				
start frame	[23]	[10]	[12]	Ours
0	0.004959	0.005027	0.004963	0.004896
100	0.012676	0.012537	0.012645	0.013198
200	0.007969	0.008093	0.008015	0.007977
300	0.009146	0.00906	0.009103	0.009078
400	0.012493	0.012359	0.012495	0.012513
500	0.009038	0.009058	0.00903	0.009121
600	0.011745	0.01158	0.011704	0.011769
700	0.011078	0.006407	0.010943	0.010734
800	0.008175	0.008196	0.008154	0.007768
900	0.009705	0.009984	0.00975	0.009786
1000	0.00686	0.00683	0.006843	0.006959
1100	0.010457	0.010102	0.010413	0.01021
1200	0.004587	0.004593	0.0046	0.004591
1300	0.016696	0.016873	0.016619	0.016349
1400	0.007154	0.006933	0.007193	0.007091
1500	0.02781	0.043692	0.028033	0.348338
1600	0.013591	0.013524	0.013501	0.012751
1700	0.00921	0.009032	0.009146	0.008781
1800	0.011597	0.011811	0.011618	0.011258
1900	0.411785	0.188297	0.511691	0.248291
2000	0.011977	0.01161	0.011919	0.011961
2100	0.008034	0.008031	0.008033	0.007999
2200	0.001378	0.001377	0.001376	0.001326
2300	0.000663	0.000668	0.000664	0.000714

*The bold character represents the minimum.

Section_offset=300				
start frame	[23]	[10]	[12]	Ours
0	0.01269	0.013046	0.012679	0.013547
300	0.01114	0.011148	0.011114	0.011107
600	0.013161	0.012968	0.013061	0.013041
900	0.011495	0.011331	0.011455	0.011332
1200	0.013392	0.013217	0.013366	0.013483
1500	0.02186	0.034901	0.021084	0.634719
1800	0.585208	0.320629	0.441459	0.153762
2100	0.005722	0.005746	0.005719	0.005722

Section_offset=500				
start frame	[23]	[10]	[12]	Ours
0	0.01699	0.016692	0.016885	0.017118
500	0.014164	0.014071	0.014062	0.0137
1000	0.014976	0.014767	0.0149	0.014927
1500	0.435192	0.204898	0.410075	0.714125
2000	0.012481	0.012785	0.01248	0.012969

Section_offset=1000				
start frame	[23]	[10]	[12]	Ours
0	0.020347	0.020131	0.020214	0.020174
1000	0.796527	0.427849	0.41353	0.638722
2000	0.012481	0.012785	0.01248	0.012969

Sequence ID 2:

Section_offset=100				
start frame	[23]	[10]	[12]	Ours
0	0.015841	0.016695	0.015791	0.015894
100	0.011803	0.011757	0.011771	0.011536
200	0.007522	0.007492	0.007466	0.007074
300	0.006901	0.007072	0.006804	0.007212
400	0.007738	0.007256	0.007643	0.007318
500	0.015179	0.010483	0.014669	0.01258
600	0.011429	0.011362	0.011295	0.009836
700	0.006864	0.006816	0.006872	0.006323
800	0.005	0.00487	0.004966	0.004829
900	0.019705	0.020002	0.019811	0.017214
1000	0.012042	0.01169	0.013324	0.011641
1100	0.015411	0.019402	0.015734	0.01531
1200	0.009692	0.009506	0.009611	0.009347
1300	0.008763	0.008922	0.008749	0.008203
1400	0.009417	0.00929	0.009387	0.007768
1500	0.007633	0.007675	0.007581	0.007729
1600	0.006087	0.005913	0.006082	0.006007
1700	0.006077	0.005889	0.006081	0.006252
1800	0.026751	0.019302	0.02676	0.032832
1900	0.004669	0.004708	0.004627	0.004451
2000	0.014922	0.013697	0.014608	0.012281
2100	0.021314	0.022167	0.021149	0.021573
2200	0.016921	0.016991	0.016817	0.017435
2300	0.006192	0.006189	0.00614	0.005775
2400	0.005222	0.005269	0.005194	0.005261

*The bold character represents the minimum.

Section_offset=300				
start frame	[23]	[10]	[12]	Ours
0	0.016522	0.017755	0.016448	0.016729
300	0.015052	0.012975	0.014888	0.015224
600	0.013188	0.013109	0.013053	0.011171
900	0.03743	0.039136	0.03768	0.032204
1200	0.014018	0.014437	0.013937	0.013977
1500	0.009857	0.009928	0.009859	0.009476
1800	0.026239	0.023084	0.025969	0.027064
2100	0.021141	0.021733	0.020977	0.021739
2400	0.005222	0.005269	0.005194	0.005261

Section_offset=500				
start frame	[23]	[10]	[12]	Ours
0	0.02486	0.023234	0.024346	0.024294
500	0.012822	0.011992	0.012739	0.011701
1000	0.016991	0.017532	0.016918	0.015946
1500	0.017743	0.017271	0.017697	0.016831
2000	0.02055	0.020624	0.020465	0.020425

Section_offset=1000				
start frame	[23]	[10]	[12]	Ours
0	0.211349	0.021918	0.29865	0.023587
1000	0.046001	0.027531	0.047702	0.070297
2000	0.02055	0.020624	0.020465	0.020425

Sequence ID 3:

*The bold character represents the minimum.

Section_offset=100				
start frame	[23]	[10]	[12]	Ours
0	0.003829	0.003521	0.003753	0.003792
100	0.005743	0.005169	0.005625	0.002812
200	0.004401	0.003317	0.00427	0.002285
300	0.010847	0.010726	0.010861	0.009246
400	0.004715	0.004697	0.004765	0.003146
500	0.004706	0.004605	0.004635	0.003384
600	0.004541	0.004891	0.004534	0.00435
700	0.009672	0.009376	0.009938	0.005435
800	0.005192	0.004615	0.005027	0.004191
900	0.005611	0.005186	0.005538	0.004352
1000	0.009277	0.008697	0.009213	0.008712
1100	0.010027	0.009615	0.009891	0.008541
1200	0.00378	0.003446	0.003675	0.003224
1300	0.002999	0.002857	0.002988	0.003693
1400	0.002805	0.002932	0.00282	0.002979
1500	0.002437	0.001909	0.002365	0.002378
1600	0.002477	0.002405	0.002436	0.00259
1700	0.006708	0.006166	0.006631	0.007373
1800	0.005839	0.005351	0.005736	0.006149
1900	0.001887	0.001818	0.001868	0.001968
2000	0.001542	0.00164	0.001508	0.001724
2100	0.001459	0.001424	0.00144	0.001472
2200	0.002807	0.002778	0.002787	0.002611
2300	0.001586	0.001341	0.001581	0.001468
2400	0.003273	0.003159	0.003271	0.003373
2500	0.002674	0.002621	0.002668	0.002786
2600	0.004675	0.004651	0.004639	0.004496
2700	0.005307	0.005142	0.005202	0.004923
2800	0.004125	0.003921	0.00411	0.004193
2900	0.00389	0.003829	0.003894	0.004035
3000	0.006285	0.006058	0.006143	0.006057
3100	0.003912	0.00359	0.003815	0.00348
3200	0.004012	0.003861	0.003947	0.004212
3300	0.001489	0.001367	0.00149	0.001395
3400	0.002121	0.001871	0.002098	0.002316
3500	0.00217	0.001989	0.002149	0.002009
600	0.000663	0.000586	0.000639	0.000659

Section_offset=300				
start frame	[23]	[10]	[12]	Ours
0	0.01508	0.013643	0.015216	0.007307
300	0.012899	0.012514	0.01304	0.010721
600	0.006944	0.0079	0.007602	0.006315
900	0.023846	0.022527	0.023531	0.021886
1200	0.008123	0.00869	0.00817	0.009266
1500	0.004402	0.004409	0.004365	0.004634
1800	0.005552	0.005185	0.005499	0.0055
2100	0.004133	0.003874	0.004134	0.005028
2400	0.005129	0.005226	0.005067	0.005708
2700	0.008889	0.008306	0.008749	0.00821
3000	0.006142	0.005869	0.006048	0.006026
3300	0.002348	0.001959	0.002304	0.002227
3600	0.000663	0.000586	0.000639	0.000659

Section_offset=500				
start frame	[23]	[10]	[12]	Ours
0	0.020726	0.019333	0.020881	0.012834
500	0.007062	0.006982	0.007058	0.006542
1000	0.016902	0.016289	0.016722	0.015829
1500	0.012671	0.011594	0.012436	0.012152
2000	0.005727	0.00515	0.005669	0.006679
2500	0.010645	0.009565	0.010455	0.009811
3000	0.005446	0.005284	0.005382	0.005667
3500	0.002168	0.001982	0.00214	0.002026

Section_offset=1000				
start frame	[23]	[10]	[12]	Ours
0	0.018206	0.01789	0.018213	0.015227
1000	0.016903	0.016661	0.016798	0.0162
2000	0.011971	0.010695	0.011707	0.011119
3000	0.005107	0.004959	0.005046	0.005321

Sequence ID 4:

Section_offset=100				
start frame	[23]	[10]	[12]	Ours
0	0.021776	0.021823	0.021762	0.021211
100	0.049838	0.05065	0.049917	0.050599
200	0.020648	0.021702	0.020903	0.022807
300	0.03363	0.036147	0.033316	0.035954
400	0.017382	0.016648	0.017219	0.015788
500	0.005075	0.00503	0.005059	0.005311

*The bold character represents the minimum.

Section_offset=300				
start frame	[23]	[10]	[12]	Ours
0	0.057408	0.090017	0.05597	0.071398
300	0.029404	0.030117	0.029095	0.028989

Section_offset=500				
start frame	[23]	[10]	[12]	Ours
0	0.054579	0.078293	0.053496	0.059787
500	0.005075	0.00503	0.005059	0.005311

Section_offset=1000				
start frame	[23]	[10]	[12]	Ours
0	0.053057	0.075028	0.052051	0.056653

Sequence ID 5:

*The bold character represents the minimum.

Section_offset=100				
start frame	[23]	[10]	[12]	Ours
0	0.006553	0.006621	0.006579	0.005602
100	0.008171	0.008033	0.008088	0.005674
200	0.015073	0.014082	0.014955	0.008602
300	0.012848	0.012309	0.012725	0.008755
400	0.013844	0.013716	0.013795	0.010758
500	0.004076	0.00404	0.004066	0.003884
600	0.005318	0.005189	0.005296	0.003706
700	0.007156	0.007176	0.006999	0.002615
800	0.005229	0.004798	0.005188	0.004919
900	0.009723	0.007341	0.010677	0.029052
1000	0.006951	0.003373	0.005857	0.016587
1100	0.009603	0.008735	0.009388	0.003382
1200	0.003753	0.00365	0.003776	0.002881
1300	0.005673	0.005428	0.005468	0.002093
1400	0.012366	0.010816	0.014589	0.011681
1500	0.043934	0.025804	0.04248	0.015434

Section_offset=300				
start frame	[23]	[10]	[12]	Ours
0	0.011371	0.011268	0.011136	0.009592
300	0.032447	0.028297	0.034449	0.014649
600	0.251593	0.182271	0.263072	0.008615
900	0.01421	0.009577	0.015734	0.053083
1200	0.126579	0.11556	0.110464	0.399631
1500	0.043934	0.025804	0.04248	0.015434

Section_offset=500				
start frame	[23]	[10]	[12]	Ours
0	0.01329	0.013089	0.01299	0.010828
500	0.01157	0.010389	0.00842	0.009516
1000	1.801615	2.279172	1.190368	1.314578

Section_offset=1000				
start frame	[23]	[10]	[12]	Ours
0	0.011794	0.011628	0.011554	0.010315
1000	1.803041	2.283701	1.186678	1.31262

Sequence ID 6:

*The bold character represents the minimum.

Section_offset=100					Section_offset=300				
start frame	[23]	[10]	[12]	Ours	start frame	[23]	[10]	[12]	Ours
0	0.003 766	0.003 599	0.003 687	0.002 212	0	0.005 629	0.005 305	0.005 522	0.002 982
100	0.003 35	0.003 29	0.003 333	0.002 707	300	0.007 496	0.005 395	0.006 803	0.003 324
200	0.002 956	0.002 893	0.002 922	0.003 571	600	0.003 92	0.003 71	0.003 945	0.002 062
300	0.011 496	0.007 718	0.010 277	0.004 1	900	0.195 453	0.190 793	0.208 065	0.320 689
400	0.003 856	0.003 619	0.003 861	0.003 978	1200	0.017 472	0.009 136	0.015 83	0.080 409
500	0.004 276	0.004 207	0.004 218	0.003 053	1500	0.001 579	0.001 521	0.001 559	0.001 262
600	0.002 163	0.002 088	0.002 188	0.001 287	Section_offset=500				
700	0.001 865	0.001 38	0.001 81	0.001 272	start frame	[23]	[10]	[12]	Ours
800	0.004 54	0.003 843	0.004 218	0.002 882	0	0.007 756	0.006 774	0.007 681	0.003 552
900	0.200 328	0.165 584	0.210 575	0.071 561	500	0.069 905	0.096 784	0.070 448	0.051 876
1000	0.004 842	0.005 14	0.005 545	0.002 942	1000	0.766 645	0.486 502	0.213 724	0.886 567
1100	0.005 648	0.009 634	0.006 423	0.003 545	1500	0.001 579	0.001 521	0.001 559	0.001 262
1200	0.001 826	0.002 131	0.001 853	0.000 734	Section_offset=1000				
1300	0.002 011	0.001 644	0.001 967	0.001 429	start frame	[23]	[10]	[12]	Ours
1400	0.003 188	0.003 188	0.003 19	0.003 018	0	0.007 546	0.007 369	0.007 532	0.036 653
1500	0.001 579	0.001 521	0.001 559	0.001 262	1000	0.800 19	0.501 283	0.213 416	0.895 78

References

1. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). In: *Computer Vision and Image Understanding*, vol 3. pp 346–359. doi:[10.1016/j.cviu.2007.09.014](https://doi.org/10.1016/j.cviu.2007.09.014)
2. Berger M, Tagliasacchi A, Seversky L, Alliez P, Levine J, Sharf A, Silva C (2014) State of the art in surface reconstruction from point clouds. *Eurographics 2014 - State of the Art Reports* 1(1):161–185. doi:[10.2312/egst.20141040](https://doi.org/10.2312/egst.20141040)
3. Besl PJ, McKay ND (1992) A method for registration of 3-D shapes. *IEEE Trans Pattern Anal Mach Intell* 14(2):239–256
4. Borgefors G (1986) Distance transformations in digital images. *Comput Vision Graph* 34(3):344–371
5. Canny J (1986) A computational approach to edge detection. *IEEE Trans Pattern Anal Mach Intell* 8(6):679–698
6. Curless B, Levoy M (1996) A Volumetric Method for Building Complex Models from Range Images. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM Press, pp 303–312
7. Endres F, Hess J, Engelhard N, Sturm J, Cremers D, Burgard W (2012) An evaluation of the RGB-D SLAM system. Paper presented at the International Conference on Robotics and Automation, May 2012
8. Fioraio N, Taylor J, Fitzgibbon A, Di Stefano L, Izadi S (2015) Large-scale and drift-free surface reconstruction using online subvolume registration. In: *2015 I.E. Conference on Computer Vision and Pattern Recognition (CVPR)*, June. IEEE, pp 4475–4483
9. Fischler MA, Bolles RC (1981) Random sample consensus - a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 24(6):381–395
10. Godin G, Rioux M, Baribeau R (1994) Three-dimensional registration using range and intensity information. In: *Proc. SPIE 2350, Videometrics III*, Oct. pp 279–290
11. Graham M, Zook M, Boulton A (2013) Augmented reality in urban places: contested content and the duplicity of code. *Trans Inst Br Geogr* 38(3):464–479
12. Gressin A, Mallet C, Demantké J, David N (2013) Towards 3D lidar point cloud registration improvement using optimal neighborhood knowledge. *ISPRS J Photogramm Remote Sens* 79:240–251. doi:[10.1016/j.isprsjprs.2013.02.019](https://doi.org/10.1016/j.isprsjprs.2013.02.019)
13. Gumhold S, Wang X, MacLeod R (2001) Feature extraction from point clouds. In: *Proceedings of 10th International Meshing Roundtable*. pp 293–305
14. Han J, Shao L, Xu D, Shotton J (2013) Enhanced Computer Vision With Microsoft Kinect Sensor: A Review. *IEEE Trans Cybern* 43(5):1318–1334
15. Handa A, Whelan T, McDonald J, Davison AJ (2014) A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: *2014 I.E. International Conference on Robotics and Automation (ICRA)*. IEEE, pp 1524–1531
16. 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. *Int J Robot Res* 31(5):647–663
17. Holzer S, Rusu RB, Dixon M, Gedikli S, Navab N (2012) Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 7–12 Oct. 2012. pp 2684–2689. doi:[10.1109/IROS.2012.6385999](https://doi.org/10.1109/IROS.2012.6385999)
18. Hoppe H, DeRose T, McDonald J, Stuetzle W, Duchamp T (1992) Surface reconstruction from unorganized points. In: *SIGGRAPH Comput. Graph*, vol 2. ACM, pp 71–78
19. Izadi S, Hilliges O, Molyneux D, Newcombe R, Kohli P, Shotton J, Hodges S, Freeman D, Davison S, Fitzgibbon A (2011) KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. Paper presented at the ACM Symposium on User Interface Software and Technology, Oct 2011
20. Low KL (2004) Linear least-squares optimization for point-to-plane ICP surface registration. Chapel Hill 4
21. Muja M, Lowe DG (2014) Scalable nearest neighbor algorithms for high dimensional data. *IEEE Trans Pattern Anal Mach Intell* 36(11):2227–2240
22. Nardi L, Bodin B, Zia MZ, Mawer J, Nisbet A, Kelly PHJ, Davison AJ, Lujan M, O'Boyle MFP, Riley G, Topham N, Furber S (2015) Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM. In: *2015 I.E. International Conference on Robotics and Automation (ICRA)*. IEEE, pp 5783–5790
23. Newcombe RA, Izadi S, Hilliges O, Molyneux D, Kim D, Davison AJ, Kohli P, Shotton J, Hodges S, Fitzgibbon A (2011) KinectFusion: Real-time dense surface mapping and tracking. Paper presented at the ISMAR '11: Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, Oct 2011
24. Rusu RB (2010) Semantic 3D object maps for everyday manipulation in human living environments. *Künstl Intell* 24(4):345–348. doi:[10.1007/s13218-010-0059-6](https://doi.org/10.1007/s13218-010-0059-6)

25. Sturm J, Engelhard N, Endres F, Burgard W, Cremers D (2012) A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 07. pp 573–580
26. Weber C, Hahmann S, Hagen H (2010) Sharp feature detection in point clouds. In: Proceedings of the 2010 Shape Modeling International Conference. IEEE, pp 175–186
27. Whelan T, Kaess M, Johannsson H, Fallon M, Leonard JJ, McDonald J (2015) Real-time large-scale dense RGB-D SLAM with volumetric fusion. *Int J Robot Res* 34(4–5):598–626
28. Wu J, Wang Q (2007) Feature point detection from point cloud based on repeatability rate and local entropy. In: MIPPR 2007: Automatic Target Recognition and Image Analysis; and Multispectral Image Acquisition. pp 1–10



Dong-Won Shin received his B.S. degree in electric and electronic engineering from Kumoh National Institute of Technology (KIT), Korea in 2013 and M.S. degree in information and communication engineering at the Gwangju Institute of Science and Technology (GIST), Korea in 2015. He is currently working towards his Ph.D. degree in the school of electrical engineering and computer science at GIST, Korea. His research interests include 3D computer vision, 3D reconstruction, SLAM, and AR system.



Yo-Sung Ho received the B.S. and M.S. degrees in electronic engineering from Seoul National University, Seoul, Korea, in 1981 and 1983, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, in 1990. He joined the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea, in 1983. From 1990 to 1993, he was with Philips Laboratories, Briarcliff Manor, NY, where he was involved in development of the advanced digital high-definition television system. In 1993, he rejoined the Technical Staff of ETRI and was involved in development of the Korea direct broadcast satellite digital television and high-definition television systems. Since 1995, he has been with the Gwangju Institute of Science and Technology, Gwangju, Korea, where he is currently a Professor in the Department of Information and Communications. His research interests include digital image and video coding, image analysis