IEICE TRANS. INF. & SYST., VOL.E100–D, NO.6 JUNE 2017

PAPER

Effective Indoor Localization and 3D Point Registration Based on Plane Matching Initialization

Dongchen ZHU[†a)], **Ziran XING**[††], ***Nonmembers***, **Jiamao LI**[†], ***Member***, **Yuzhang GU**[†], ***and*** **Xiaolin ZHANG**[†], ***Nonmembers***

SUMMARY Effective indoor localization is the essential part of VR (Virtual Reality) and AR (Augmented Reality) technologies. Tracking the RGB-D camera becomes more popular since it can capture the relatively accurate color and depth information at the same time. With the recovered colorful point cloud, the traditional ICP (Iterative Closest Point) algorithm can be used to estimate the camera poses and reconstruct the scene. However, many works focus on improving ICP for processing the general scene and ignore the practical significance of effective initialization under the specific conditions, such as the indoor scene for VR or AR. In this work, a novel indoor prior based initialization method has been proposed to estimate the initial motion for ICP algorithm. We introduce the generation process of colorful point cloud at first, and then introduce the camera rotation initialization method for ICP in detail. A fast region growing based method is used to detect planes in an indoor frame. After we merge those small planes and pick up the two biggest unparallel ones in each frame, a novel rotation estimation method can be employed for the adjacent frames. We evaluate the effectiveness of our method by means of qualitative observation of reconstruction result because of the lack of the ground truth. Experimental results show that our method can not only fix the failure cases, but also can reduce the ICP iteration steps significantly.

key words: *ICP (Iterative Closest Point), indoor localization, indoor reconstruction, plane matching, rotation estimation*

1. Introduction

With the trend of VR and AR, the indoor positioning technology has been the focus of attention. This problem comes from the need of tracking the HMD (Helmet Mounted Displays) in real time. More precisely, the VR or AR system need the HMD to report its orientation and location under the room coordinate system.

There are many existed technologies which can support this work, such as putting IMU (Inertial Measurement Unit) sensor in HMD, like Oculus and GearVR, or using laser based tracking system, like HTC Vive. Since IMU sensor only can report the acceleration or angular acceleration at each time, users need to integrate the raw sensor data twice to get the location and orientation. Due to the measurement error in each time, only relying on the IMU sensor will cause pose drift and inconsistent with the global frame. The laser based tracking system can archive very high accuracy in real time, but the safe area is quite small and limited by laser range.

Another famous class of sensor is camera, and there are many works focus on vision based positioning technology such as [1]–[3]. The kernel of vision method is so called SFM (Structure from Motion), which uses many pairs of corresponding points to recover camera motion. There are some well established theories to explain at what conditions that camera pose can be recovered [4], but life is not so easy, since these theories have a common assumption that point correspondences can be correctly got from two images. In indoor scene, walls and roofs become main components in the images, and correspondences are hard to find due to less texture on these surfaces [5]. To deal with such blank wall problem, people started to use depth camera to help, and the so called joint camera system is RGB-D camera. These cameras can capture color frame and depth frame at the same time, then we can recover colorful point cloud of the scene if we pre-calibrate the camera. In this work, we present a new method to track camera pose in the indoor scene and reconstruct the scene by using Kinect 2.0 as RGB-D camera.

If we can track the camera pose, we can transform two point clouds from adjacent frames into one coordinate system to get a larger point cloud with respect to the real scene structure. This problem is also known as 3D map registration, and there exists a famous algorithm ICP to solve it, which is firstly introduced by Besl et al. [6] as Eq. (1).

$$\sum_{i=1}^{n}(\boldsymbol{R}\boldsymbol{A}_i + \boldsymbol{t} - \boldsymbol{B}_i) \tag{1}$$

where $\boldsymbol{A}_i \in \mathbb{R}^{3\times1}$ is the 3D-coordinate of point i in frame A, $\boldsymbol{B}_i \in \mathbb{R}^{3\times1}$ is the 3D-coordinate of point i in frame B, $\boldsymbol{R} \in SO(3)$ and $\boldsymbol{t} \in \mathbb{R}^{3\times1}$ are the relative rotation and translation between frame A and B respectively.

This problem is a kind of 'Chicken-Egg' problem. If we have the correspondence information, we can get optimal motion estimation by minimizing Eq. (1). On the other hand, if we have coordinates transform information, we can easily find corresponding pairs by searching the closest points. The tricky part of registration problem is that finding correspondences correctly in two point clouds is not easy, just like SFM.

ICP is a kind of iterative algorithm which can bypass

Manuscript received September 5, 2016.
Manuscript revised January 20, 2017.
Manuscript publicized March 8, 2017.

[†]The authors are with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China.
[††]The author is with the ShanghaiTech University, Shanghai 201210, China.
a) E-mail: dchzhu@mail.sim.ac.cn

DOI: 10.1587/transinf.2016EDP7379

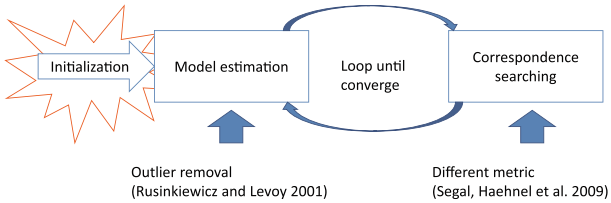


Fig. 1 The ICP workflow. It is an iterative method. If we have correspondence, we can get the transform; otherwise, if we have transform, we will have the correspondence. A good transform initialization is very helpful for good results.

(a) (b)

Fig. 2 Examples of failure cases of ICP. The mistakes are very obvious, like two blackboards in (a), two clocks in (b) and the messy ceilings in both of them.

the difficulties of correspondence searching. The main procedure of ICP is relatively simple, summarized as Fig. 1. The current motion is utilized to estimate correspondence at first, then the newly found correspondence is utilized to update motion. When motion changing is small during adjacent iteration, the ICP algorithm converges. Here we need an initialization to enable iteration, and normally people just use identity matrix for rotation and zeros for translation. This initialization assumes that the relative motion between adjacent frames is small. Under this assumption, Kinect Fusion [7] used ICP for camera tracking component. Although with some improvement works in [8], [9], zero motion initialized ICP can not work all the time. Figure 2 shows some failure cases.

Many works focused on improving ICP itself but initialization part for solving the general case. In this work, we focus on the indoor localization and reconstruction, which means we will have some room prior to make better results. Section 2 will simply introduce the sensor calibration and colorful point cloud recovery. Section 3 will introduce the plane matching based rotation initialization method for ICP. Section 4 will show some experiment results to verify the effectiveness of our method. We not only compare our method with non-initialized ICP method, but also compare our method with another plane-based method descried in [10], [11].

2. Sensor Calibration and Point Cloud Color Tracing

Tracking depth camera is very different with tracking regular camera. The relative pose of depth and color cameras is needed and the calibration method is different from two color cameras which is a mature technology. Although the intrinsic and external parameters of Kinect are precalibrated in factory, it is well known that these parameters

(a) (b)

Fig. 3 The necessity of calibration. The point cloud in (a) is formed without calibrating the Kinect, and as you can see the color of cabinet is wrong given the wall. After we calibrate our Kinect, the error is fixed, as shown in (b).

vary from devices and are not accurate enough for many applications because average parameter values of different devices was preset by factory [12]. Figure 3 shows an example point cloud before and after calibration. In this paper, we need to calibrate our Kinect to eliminate the camera parameters effects for the observation of registration results. There are many toolbox for this work like [13] and [14]. We just introduce our method briefly here and you also can use other methods to calibrate.

Since we have depth information from Kinect, we can recover the 3D coordinate of each pixel by Eq. (2), in other words, we can get the 3D point cloud of this frame.

$$\begin{pmatrix} X_D \\ Y_D \\ Z_D \end{pmatrix} = \text{Depth}\begin{pmatrix} u_D \\ v_D \\ 1 \end{pmatrix} \cdot \boldsymbol{K}_D^{-1} \cdot \begin{pmatrix} u_D \\ v_D \\ 1 \end{pmatrix} \tag{2}$$

where $\begin{pmatrix} X_D \\ Y_D \\ Z_D \end{pmatrix}$ represents the 3D point coordinates in depth camera frame, $\begin{pmatrix} u_D \\ v_D \\ 1 \end{pmatrix}$ is the homogenous depth image coordinates of projected points, $\text{Depth}\begin{pmatrix} u_D \\ v_D \\ 1 \end{pmatrix}$ means the depth value of $\begin{pmatrix} u_D \\ v_D \\ 1 \end{pmatrix}$ and $\boldsymbol{K}_D$ is the intrinsic matrix of depth camera.

According to Eq. (2), we need to know the intrinsic matrix $\boldsymbol{K}_D$, which means we should calibrate the intrinsic parameters of the depth camera. Although color data is not been directly used for tracking the sensor in this work, the final result is compared by 3D map alignment result due to the lack of the ground truth. This criterion is reasonable because if we could align two point clouds perfectly, the camera motion must be accurate. For visualization the alignment result, the intrinsic of color camera and the extrinsic between color and depth camera must be calibrated too.

2.1 Camera Intrinsic Calibration

The most famous calibration method is Zhang's method [15].

This method uses checkerboard as calibration target. By taking several pictures of the pattern, the camera intrinsics can be easily estimated with detecting the corner points of the board. It is very useful for color camera but things are not so easy for depth one, since the output is just the depth map and corner points can not be detected anymore. Some methods used special designed calibration target. By knowing the geometry, the intrinsics can be recovered. Although these methods sound great, in practice, building such accurate calibration target is not easy or sometimes expensive.

By knowing the facts of Kinect depth map generation process, the calibration method can be fairly easy. The Kinect 2.0 measures depth by ToF (Time of Flight) sensor, the basic principle of this sensor is just like laser scanner. This kind of sensor projects some light (maybe can not be seen by human eyes) and measures the return time of the light to recover depth. For Kinect sensor the projected light is infrared, if we use black color and not specular surface, the light will be absorbed by material and never reflect. Based on this idea, we used the Zhang's method for camera intrinsic calibration by access the raw infrared images from the Kinect.

2.2 Extrinsic Calibration and Color Tracing

Before we trace the color information for point cloud, we must know the relative orientation and translation between color and depth. The calibration method is just like stereo calibration method. By considering color and depth cameras as rigid stereo pair, we computed the depth and color extrinsic by a common pairs.

Once we have the extrinsic, we can simply project the point cloud into color camera's image plane by Eq. (3) and Eq. (4). For any projected points which is in inside of the image region, we can use the color information specified by pixel coordinates of projected points. Figure 4 shows an example of one frame result.

$$\begin{pmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \cdot \begin{pmatrix} X_D \\ Y_D \\ Z_D \\ 1 \end{pmatrix} \tag{3}$$

$$\lambda \begin{pmatrix} u_C \\ v_C \\ 1 \end{pmatrix} = \boldsymbol{K}_C \cdot \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} \tag{4}$$

where $\begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix}$ represents the 3D point coordinates in color camera frame, $\begin{pmatrix} X_D \\ Y_D \\ Z_D \end{pmatrix}$ represents the 3D point coordinates in depth camera frame, $\begin{pmatrix} u_C \\ v_C \\ 1 \end{pmatrix}$ is the homogenous color image coordinates of projected points, $\boldsymbol{K}_C$ is the intrinsic matrix of color camera, $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation and translation, λ is the normalization factor.

Fig. 4 Example of one frame point cloud.

3. Plane Matching Based Rotation Estimation

As expressed in introduction section, this work focused on how to estimate a good initialization for ICP based on the indoor scene prior. The main idea is the rotation can be recovered if we can detect two unparallel walls and correctly matching them in adjacent frames. This section will present how to detect and match two planes in adjacent frames, and then get a rotation initialization for ICP.

3.1 Detect Planes in Point Cloud

The most common structure of the room is wall. In most cases, the surface of the wall can be viewed as static planar scene. If we can track the wall, we will get a accurate and consistent result since we know wall is planar and cannot move. Especially, for dynamic scene, tracking walls are extremely useful since tracking dynamic object will cause wrong result owing to the changed reference frame.

Classical plane detection method is RANSAC (RANdom Sample Consensus) based [16]. This method is powerful when we want fit one plane in point cloud. But in this work, we need extract all the planar scenes and statistically analyse those to extract unparallel large planes as the feature of this point cloud.

In this work, a fast region growing based method [17] has been implemented as shown in Algorithm 1. The parameters $(\delta, \epsilon, \gamma, \theta)$ are the threshold given by ourselves. Ω is the optimal plane of the plane $\boldsymbol{P} \bigcup \boldsymbol{p'}$. The $\mathbb{V}$ is the set of accessed points. The $d(A, B)$ represents the distance measurement between A and B.

By incrementally compute MSE (Mean Square Error) which defined how likely this region belongs to a plane and covariance matrix which is used to get plane equation, the running time cost of adding a new point into a region can be reduced from $O(n)$ to $O(1)$ if point cloud size is n. Thanks to the priority queue, the nearest neighbor searching cost can be reduced from $O(n)$ to $O(\log n)$. The overall running time

Algorithm 1 Planes identification.

Input: Point cloud, **PC**;
Output: Detected plane set, $\mathbb{P}$;
Initialize: $\mathbb{P} \leftarrow \emptyset$; $\mathbb{V} \leftarrow \emptyset$;
while Exist two points $p_1, p_2 \in$ **PC** $- \mathbb{V}$ **do**
  plane $\boldsymbol{P} \leftarrow \{p_1, p_2\}$;
  while new point can be found **do**
    select nearest neighbor point $\boldsymbol{p}'$ such that $d(\boldsymbol{P}, \boldsymbol{p}') < \delta$;
    if $(\mathtt{MSE}(\boldsymbol{P} \bigcup \boldsymbol{p}') < \epsilon$ and $d(\Omega, \boldsymbol{p}') < \gamma)$ **then**
      $\boldsymbol{P} \leftarrow \boldsymbol{P} \bigcup \boldsymbol{p}'$;
    end if
  end while
  if $(\mathtt{size}(\boldsymbol{P}) > \theta)$ **then**
    $\mathbb{P} \leftarrow \mathbb{P} \bigcup \boldsymbol{P}$;
  end if
  $\mathbb{V} \leftarrow \mathbb{V} \bigcup \boldsymbol{P}$
end while
return $\mathbb{P}$;

can be reduced to $O(n \log n)$.

3.2 Merge Small Planes

Once we have extracted those planes, the statistically analyse should be performed on them to get the most representative ones. The method used in this work is a simple K-cluster algorithm. The merge algorithm is summarised in Algorithm 2. This algorithm merges two small planes which have similar plane normal and bias until there do not exist any similar planes.

Algorithm 2 Planes merging.

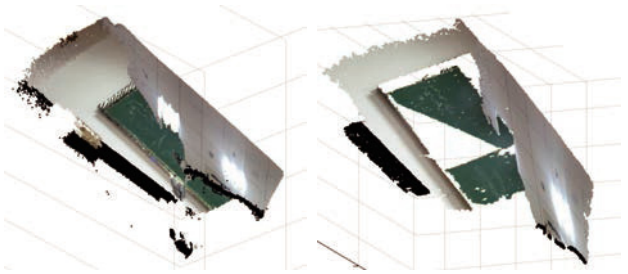
Input: Detected plane set, $\mathbb{P}$;
Output: Merged plane set, $\mathbb{P}'$;
Initialize: $\mathbb{P}' \leftarrow \mathbb{P}$;
while Exist two planes $\boldsymbol{P}_1, \boldsymbol{P}_2 \in \mathbb{P}'$ such that $d_n(\boldsymbol{P}_1, \boldsymbol{P}_2) < T_{\text{angle}}$ and $d_b(\boldsymbol{P}_1, \boldsymbol{P}_2) < T_{\text{bias}}$ **do**
  $\widetilde{\boldsymbol{P}} = \text{merge}\{\boldsymbol{P}_1, \boldsymbol{P}_2\}$;
  $\mathbb{P}' = \mathbb{P}' - \{\boldsymbol{P}_1, \boldsymbol{P}_2\}$;
  $\mathbb{P}' = \mathbb{P}' + \widetilde{\boldsymbol{P}}$
end while
return $\mathbb{P}'$;

A plane can be described by $\boldsymbol{n} \cdot \boldsymbol{p} + \mathtt{bias} = 0$, in which $\boldsymbol{n}$ is the plane normal and $\boldsymbol{p}$ is the plane point. In this paper, the plane normal we mentioned is the unit vector by default. The angle similarity used in this paper is defined by $|\boldsymbol{n}_a \cdot \boldsymbol{n}_b|$, where $\boldsymbol{n}_a$ and $\boldsymbol{n}_b$ represent plane normals of plane a and b respectively. The threshold T_{angle} and T_{bias} are given by ourselves.

After we merged the planes, we should pick out two non-parallel planes for initialization. By assuming wall is a large plane, we simply chose two biggest planes from the merged planes. A detection example is shown in Fig. 5.

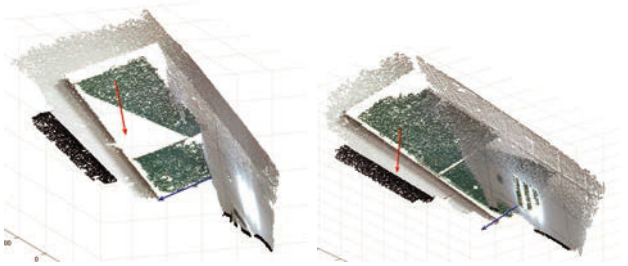
3.3 Plane Matching Based Rotation Estimation

From previous parts, we have got two main plane normals in each frame. Then we should match them between adjacent



(a) (b)

Fig. 5 A example of the two biggest planes detection. (a) is the original point cloud, and (b) is the two biggest planes picked out by our merge algorithm.



(a) (b)

Fig. 6 Example of plane matching. (a) and (b) are the main planes from two adjacent frames. According to our calculation, the red normal vector is $(0.3088, 0.2709, 0.9118)^T$ and the blue one is $(0.0287, -0.9663, 0.2558)^T$ in (a). In the same way, we know the red one is $(0.1459, 0.2316, 0.9618)^T$ and the blue one is $(0.0557, -0.9660, 0.2524)^T$ in (b). Based on the data, we can get the right rotation estimation using the Algorithm 3.

frames. Denote two plane normals in frame i as $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$. Similarly, denote two plane normals in frame $i+1$ as $\widetilde{\boldsymbol{n}_1}$ and $\widetilde{\boldsymbol{n}_2}$. Our problem is to find the matching pairs $(\boldsymbol{n}_1 = \widetilde{\boldsymbol{n}_1}, \boldsymbol{n}_2 = \widetilde{\boldsymbol{n}_2})$ or $(\boldsymbol{n}_1 = \widetilde{\boldsymbol{n}_2}, \boldsymbol{n}_2 = \widetilde{\boldsymbol{n}_1})$.

Figure 6 shows a plane matching example in adjacent frames. Since there only exist two kinds of matching and based on our assumption the motion between adjacent frames is small, so we can just explore these two possible choices and find the smallest change result. If $(\boldsymbol{n}_1 = \widetilde{\boldsymbol{n}_1}, \boldsymbol{n}_2 = \widetilde{\boldsymbol{n}_2})$, the rotation estimation algorithm can be summarized as Algorithm 3. We also can get the rotation when $(\boldsymbol{n}_1 = \widetilde{\boldsymbol{n}_2}, \boldsymbol{n}_2 = \widetilde{\boldsymbol{n}_1})$. Then two rotation angles are compared and the rotation matrix with smaller angle is what we want based on our small motion assumption.

Algorithm 3 Rotation estimate.

Input: $(\boldsymbol{n}_1, \boldsymbol{n}_2)$ and $(\widetilde{\boldsymbol{n}_1}, \widetilde{\boldsymbol{n}_2})$;
Output: rotation matrix, $\boldsymbol{R}$;
$\boldsymbol{n}_3 = \frac{\boldsymbol{n}_1 \times \boldsymbol{n}_2}{\|\boldsymbol{n}_1 \times \boldsymbol{n}_2\|}$;
$\widetilde{\boldsymbol{n}_3} = \frac{\widetilde{\boldsymbol{n}_1} \times \widetilde{\boldsymbol{n}_2}}{\|\widetilde{\boldsymbol{n}_1} \times \widetilde{\boldsymbol{n}_2}\|}$;
$\boldsymbol{R}' = [\widetilde{\boldsymbol{n}_1}, \widetilde{\boldsymbol{n}_2}, \widetilde{\boldsymbol{n}_3}] \cdot [\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3]^{-1}$;
$[\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}] = \text{svd}(\boldsymbol{R}')$;
$\boldsymbol{R} = \boldsymbol{U} \cdot \boldsymbol{V}'$
return $\boldsymbol{R}$;

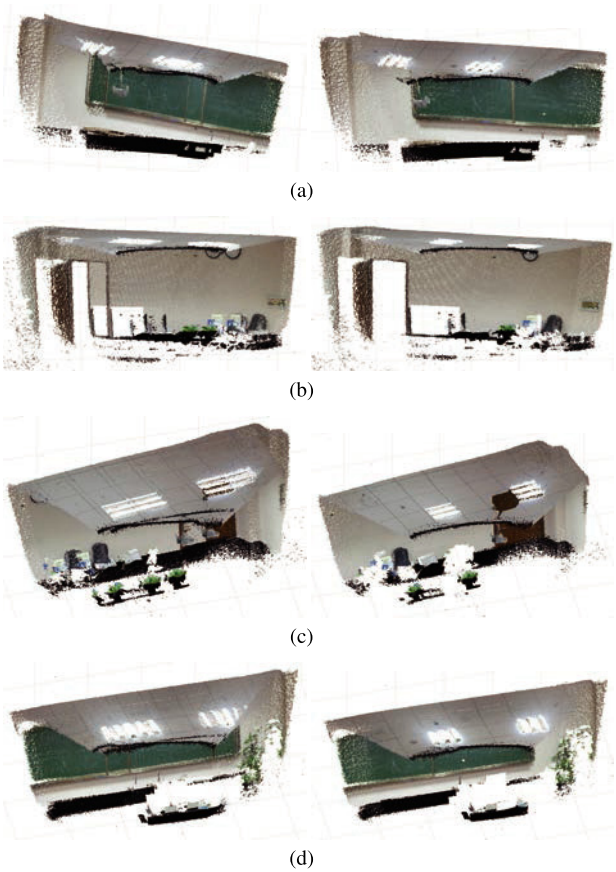


Fig. 7 Compare of results. The left one of each sub-figure is the result without initialization and the right one is initialized with our method. Easily observed, with our initialization, the results of ICP are greatly improved.

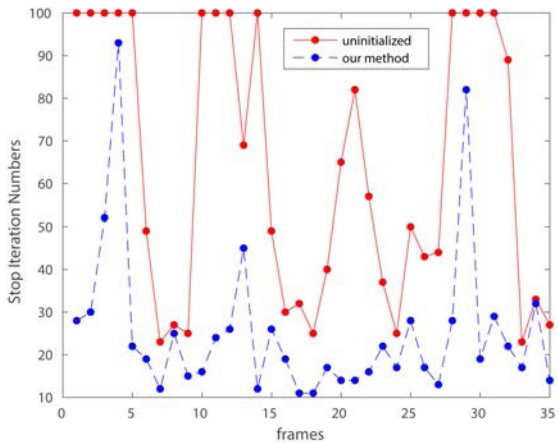


Fig. 8 Statistics of iteration numbers. Besides increasing the chances of archiving success, my method can also reduce the iteration numbers, especially for failure cases.

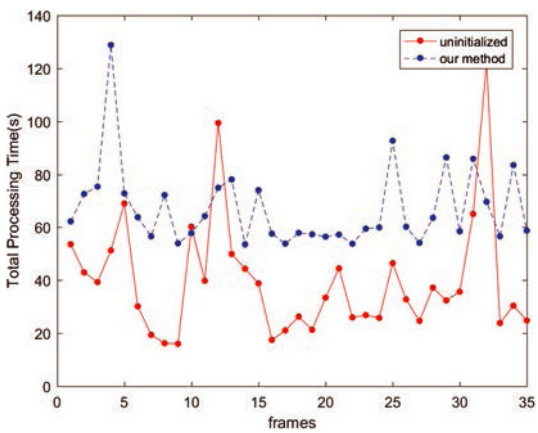


Fig. 9 Statistics of processing time for two adjacent frames. Although our method basically cost more time than non-initialized ICP for one pair point cloud, the total processing time for 36 frames can be reduced to 25 min for our method with right results, while the non-initialized ICP method costs 23 min and 16 of 35 results are obvious wrong.

4. Experiment

In this work, we used Kinect 2.0 as RGB-D sensor and took 36 color and depth frames from a room. According to the method expressed in Sect. 2, non-color point clouds are generated from depth map by Eq. (2). Then using the method in Sect. 2.2, we can get the color for each point in each frame. With the method in Sect. 3, the initialized ICP can be done for each adjacent frame pairs. In fact, we can get the camera tracking result and the room's 3D reconstruction result at the same time. Due to the lack of the ground truth, the result is verified by 3D map registration result for qualitative observation.

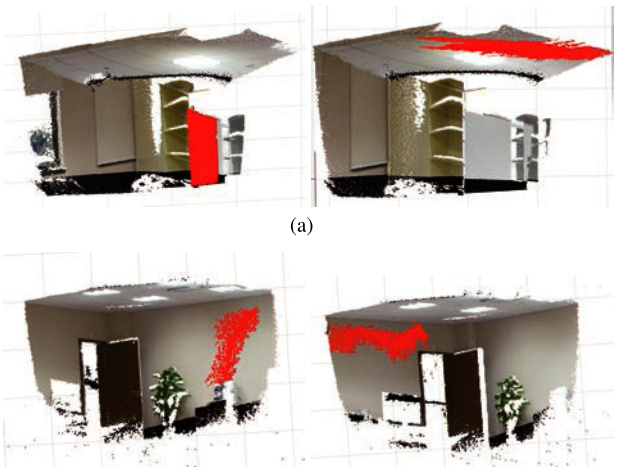
4.1 Compare with Non-Initialized ICP

In this part, we compare the results of our initialized ICP with the non-initialized one, especially for those failure cases without initialization, shown as Fig. 7. No doubt about that our method is very helpful to fix those cases. The difficult wall issue is well handled now. In fact, we also compared the non-initialization success cases with our results. Without ground truths to evaluate quantitatively, it is hard to estimate the improvement of our method for success cases. However, the qualitative observation shows that we did not make it worse.

On the other hand, we did statistics of the stop iteration numbers of those two method to evaluate quantitatively. The result is shown in Fig. 8. For all worked frames, our initialization method can reduce the iteration times obviously, particularly for those failure cases. We did the experiments with limiting the maximal number of iterations as 100. Most of the failure cases done 100 times iteration as shown in Fig. 8, but still can not get good results. However, our method need less iteration times and can get better results, like those examples in Fig. 7. Our method lets ICP more efficiently and effectively.

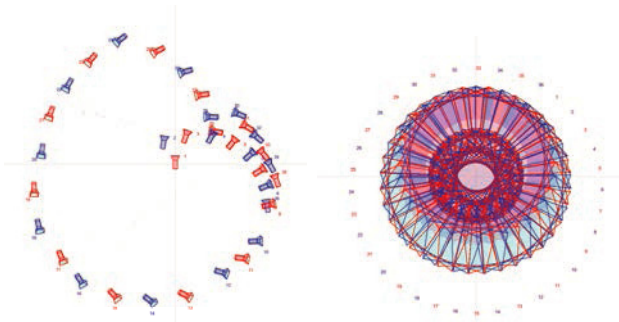
Furthermore, we discuss the trade-off between processing time and performances of using the initialization step. We do the experiments with MATLAB on a computer with a dual-core Intel Core i7-6650U CPU and 16GB RAM. Only CPU is used, and we did not use the GPU to speed up the

process here. As you probably guess, for a registration problem with two adjacent frames, additional initialization step may easily cause the increment of total processing time, as shown in Fig. 9. In this figure, we calculated the processing time from inputting the point clouds to outputting the camera poses for each adjacent frame pair. To be more specific, camera poses are directly calculated from point clouds for non-initialized ICP method, so we just need calculate this time. However, for our method, the time consists of three parts: the time of plane detection and merger for two adjacent frames, the time of rotation estimation and the time of



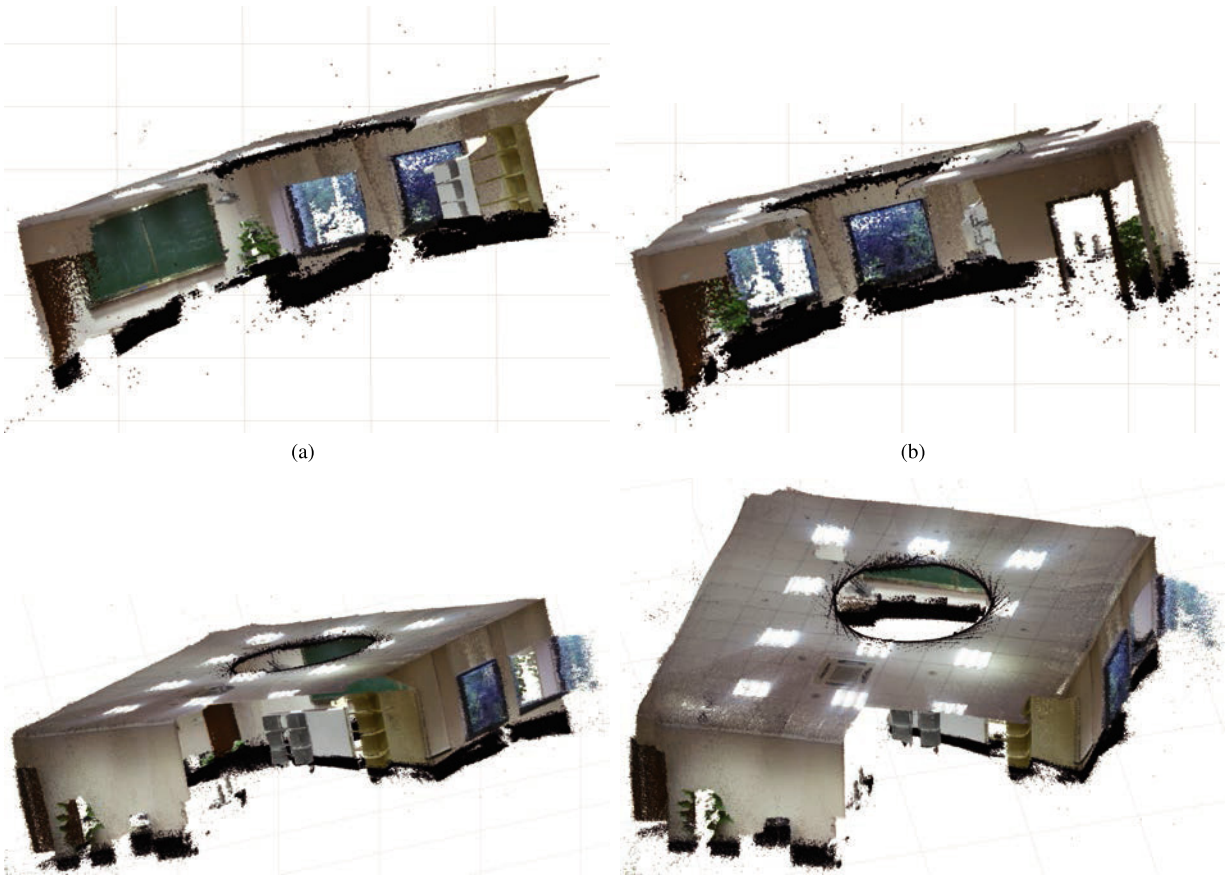
(a) (b)

Fig. 10 The wrong correspondences found by [10], [11]'s method. Each sub-figure comes from adjacent frame pairs. We color the plane pairs red, so the mistakes can be observed intuitively. In (a), a wall and a ceiling are detected, and different walls are detected in (b).



(a) (b)

Fig. 11 The camera positions. (a) shows the camera positions of the 36 frames from the room. If we ignore the translation and just observe the rotation of the camera, we can get (b).



(a) (b) (c) (d)

Fig. 12 Whole room map registration. Using 36 color and depth frames from a room, we can recover the loop trajectory ((a) and (b) show parts of the room) and produce the whole room map ((c) and (d)).

ICP iteration.

In Fig. 9, our method basically cost more time than non-initialized ICP, because detecting and merging planes for one frame cost about 23s~25s in our experiment and double time is needed for one pair. However, if plane detection and merger are done for each frame only once and then the results are saved, the total processing time for 36 frames can be reduced from 39 min to 25 min with right results. However, the non-initialized ICP method costs 23 min and 16 of 35 results are obvious wrong. In our view, increasing 2 minutes to improve the success rate is worthy. Moreover, with the enlargement of the question scale, our method will show more advantages.

Fig. 13 The limitation of our method. When sensor turning around corners, the tracked plane may shift to a new one. We color the planes red for viewing.

4.2 Compare with Another Plane-Based Method

As the important feature of indoor sense, plane feature is widely considered in many applications like localization and reconstruction. In this part, we compare the results of our initialized ICP with another plane-based method [10], [11], which also finds the correspondence of planes and then estimates the camera pose. Our method merges small planes and aims to find the two biggest planes which we regard them as reliable references, however, in [10], [11], they do not merge small planes and try to find plane correspondences which maximizes the overall geometric consistency.

Although [10], [11] have excellent results in their experiments, we find it not robust enough in simple indoor scene without so many planes, like our classroom data. [10], [11]'s success depends primarily on finding the first pair of corresponding planes. However, for our 35 pairs of frames point cloud, 20 pairs of them fail at this step of corre-

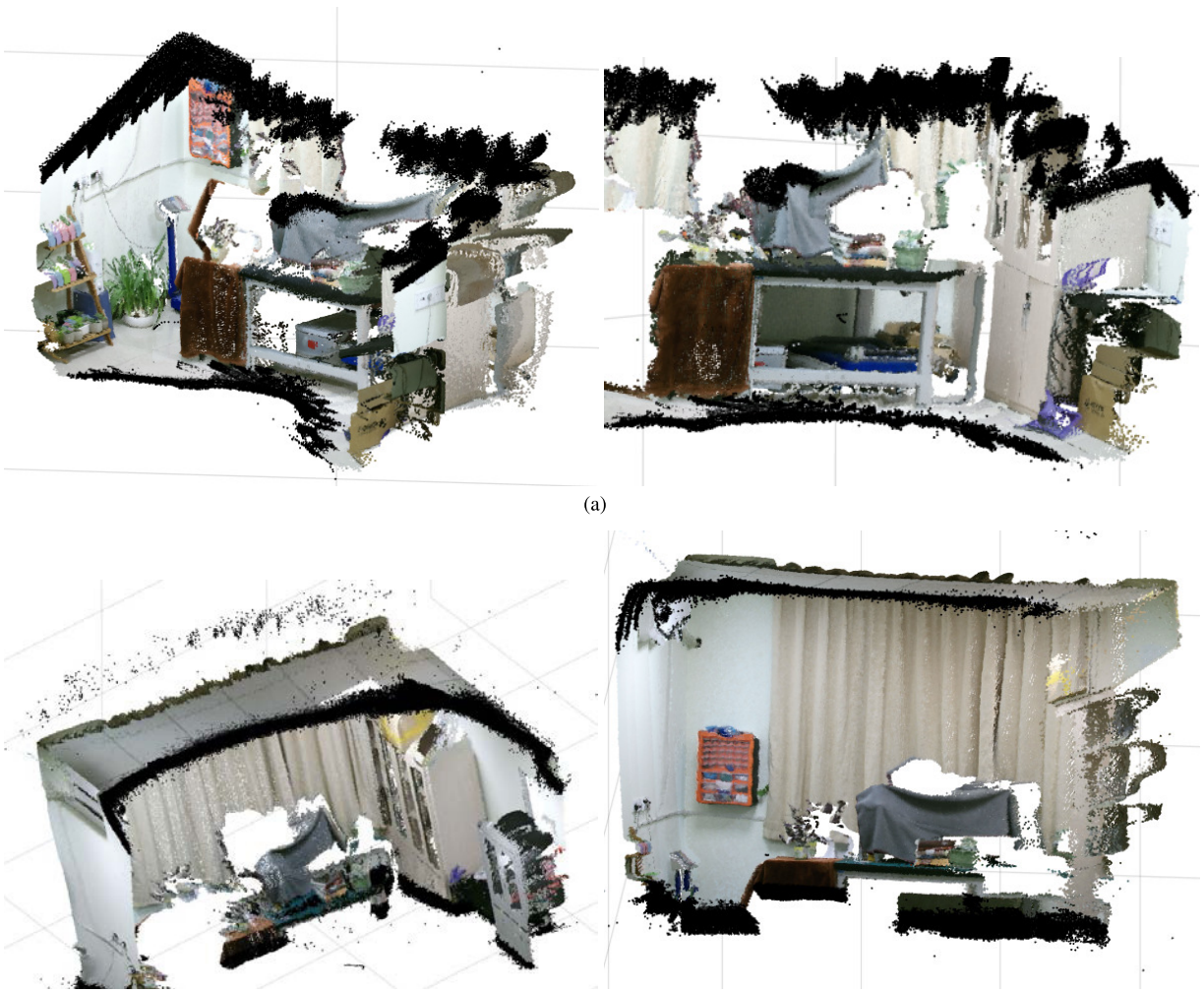


Fig. 14 The registration results of other datasets.

spondences finding. A wall and a ceiling are wrong detected as a correspondence shown as Fig. 10 (a) for example. And different orientated walls also are wrong detected, shown as Fig. 10 (b), which can not be avoided even using the color information to check because they are very similar. In addition, 7 of the remaining 15 frame pairs cannot estimate the rotation since there are no two nonparallel pairs of planes detected. Due to the lack of ground truth, we do not compare the accuracy of rest registration results here, but this method is with obvious shortages.

4.3 Loop Trajectory Positioning and Whole Room Map Registration

After we got the transformation between each adjacent frame pairs, the loop trajectory can be obtained naturally, as Fig. 12 and Fig. 11.

Because of our accurate advantages of transformation estimation, the cumulative error of the whole trajectory positioning is relatively small. In our experiments, this point reflects more obvious. We took 36 color/depth frames from a room which around a circle with about ten degrees intervals, and the whole room map registration can be done as you can see in Fig. 12 (c) and Fig. 12 (d), where no obvious mistakes occur in the end of the circle.

The camera positions can be draw as Fig. 11 (a). Since we just give a method to estimate the rotation, the translation affects the results more or less. If we just draw the rotation of the camera, we can get the Fig. 11 (b) which is approximated to the angle changes of data collection process intuitively.

5. Conclusion

In this article, a novel initialization method for ICP is presented. The initialization method is based on the indoor scene prior with a small motion assumption, and uses plane matching for rotation initialization. Although this method works well in most cases, when sensor turning around corners, plane matching result may not be reliable since original tracked plane shift to a new one, shown as Fig. 13. By assuming small motion of the sensor, once newly estimated rotation angle is much different with previous result, the previous result will be used for current initialization. Combine all methods introduced before, the loop trajectory can be recovered and the whole room map can be reconstruction as Fig. 12 (c) and Fig. 12 (d). We also applying our method on other datasets, and some registration results are shown in Fig. 14.

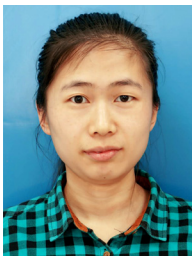
In this work, color information is only used for verify the tracking result rather than help tracking sensor. The next step work will try to add color information to get more accurate and robust result. Besides, the translation initialization and cumulative error problem will be considered in the future work.

Acknowledgments

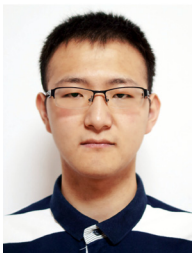
This work is partly supported by the Science and Technology Commission of Shanghai Municipality (Grant 14YF1407300).

References

[1] B. Williams, G. Klein, and I. Reid, "Real-time slam relocalisation," In IEEE International Conference on Computer Vision, pp.1–8, 2007.
[2] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," In IEEE International Conference on Computer Vision, pp.1449–1456, 2013.
[3] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," Springer International Publishing, vol.8690, pp.834–849, 2014.
[4] Y. Ma, S. Soatto, J. Košecká, and S.S. Sastry, An Invitation to 3-D Vision: From Images to Geometric Models, Springer, 2005.
[5] S. Ikehata, H. Yang, and Y. Furukawa, "Structured indoor modeling," In IEEE International Conference on Computer Vision, pp.1323–1331, 2015.
[6] P.J. Besl and N.D. Mckay, "Method for registration of 3-d shapes," IEEE Trans. Pattern Anal. Mach. Intell., vol.14, no.2, pp.239–256, 1992.
[7] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," In ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, pp.559–568, Oct. 2011.
[8] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," In 3DIM, pp.145–152, 2001.
[9] A. Segal, D. Hähnel, and S. Thrun, "Generalized-ICP," In Robotics: Science and Systems V, University of Washington, Seattle, USA, June 28–July 1, 2009.
[10] K. Pathak, N. Vaskevicius, J. Poppinga, M. Pfingsthorn, S. Schwertfeger, and A. Birk, "Fast 3D mapping by matching planes extracted from range sensor point-clouds," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, pp.1150–1155, 2009.
[11] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation," Journal of Field Robotics, vol.27, no.1, pp.52–84, 2010.
[12] C. Raposo, J.P. Barreto, and U. Nunes, "Fast and Accurate Calibration of a Kinect Sensor," 2013 International Conference on 3D Vision - 3DV 2013, Seattle, WA, pp.342–349, 2013.
[13] D. Herrera C., J. Kannala, and J. Heikkilä, "Joint Depth and Color Camera Calibration with Distortion Correction," in IEEE Trans. Pattern Anal. Mach. Intell., vol.34, no.10, pp.2058–2064, Oct. 2012.
[14] N. Burrus, Kinect calibration, http://nicolas.burrus.name/index.php/Research/KinectCalibration, 2010.
[15] Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol.22, no.11, pp.1330–1334, 2000.
[16] P.H.S. Torr, A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," Computer Vision & Image Understanding, vol.78, no.1, pp.138–156, 2000.
[17] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," In Ieee/rsj International Conference on Intelligent Robots and Systems, pp.3378–3383, 2008.



Dongchen Zhu received the B.S. degree in Wuhan University, Wuhan, China, in 2013. She is currently working towards the Ph.D. degree in Information and Communication Engineering in Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China. Her current research includes computer vision, image processing and deep learning.



Ziran Xing received the B.S. degree in North University of China, Taiyuan, China, in 2013. He is currently working towards the Ph.D. degree in ShanghaiTech University, Shanghai, China. His current research focuses on computer vision.

Jiamao Li is an associate professor of Shanghai institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China. He received his Ph.D. degree in Tokyo Institute of Technology. His research focuses on computer vision, biomedical engineering and visual physiology.

Yuzhang Gu is an associate professor of Shanghai institute of Microsystem and Informa tion Technology, Chinese Academy of Sciences, Shanghai, China. He received his Ph.D. degree in Tokyo Institute of Technology. His research focuses on computer vision, target tracking, object detection and 3D video.

Xiaolin Zhang is a professor of Shanghai institute of Microsystem and Information Tech nology, Chinese Academy of Sciences, Shang hai, China. He received his Ph.D. degree in- Yokohama National University. His research fo cuses on visual physiology, robotics, image processing and computer vision.