Accurate Camera Pose Estimation for KinectFusion Based on Line Segment Matching by LEHF

Yusuke Nakayama*, Toshihiro Honda* and Hideo Saito*
*Graduate School of Science and Technology
Keio university, Japan
Email:{*nakayama, t-honda, saito*}*@hvrl.ics.keio.ac.jp*

Masayoshi Shimizu† and Nobuyasu Yamaguchi †
†Fujitsu Laboratories Ltd.
Email:{*shimizu.masa, nobuyasu*}*@jp.fujitsu.com*

***Abstract*—KinectFusion is able to build a 3D reconstruction in real time and provide a 3D model. KinectFusion uses Iterative Closest Point (ICP) algorithm for point cloud alignment from the each camera frame and estimates each camera pose. However, ICP algorithm has its limits and the camera poses lack in accuracy. We propose an alignment method which is not only based on point cloud but also line segments. This method significantly improve the camera pose accuracy obtained from KinectFusion and creates better 3D model. In this method, we use line segment matching by Line-based Eight-directional Histogram Feature(LEHF). We also propose an improved version of LEHF for this alignment method. The basic idea is to get a set of 2D-3D line segment correspondences between 2D line segments on camera images and 3D line segments of 3D line segment based models, to solve the PnL problem and to recompute the camera pose. The experimental result that the camera pose estimated by our method is more accurate than the original one obtained from KinectFusion.**

I. Introduction

3D reconstruction has been a hot topic in Computer Vision for over three decades. Researches on 3D reconstruction are very useful for many applications, such as augmented reality (AR) system or self-location estimation and understanding of environments for robots. Structure from motion (SFM) and multi-view stereo (MVS) are some of the researches related to 3D reconstruction. *Photo explorer*[10] which reconstructs 3D points and viewpoints from collections of photographs uses SFM approach. Several algorithms about MVS are introduced in [9]. These methods require computation time. Methods of real-time 3D reconstruction have recently seen great progress. Among many online reconstruction methods, KinectFusion[7] is widely used. KinectFusion uses the Kinect and GPU algorithms to estimate the camera pose and build a dense scene reconstruction in real time. To achieve 3D model of the scene, KinectFusion system continually tracks camera pose and fuses live depth data from the camera into a single global 3D model in real-time. In the camera tracking phase, the Iterative Closest Point (ICP) algorithm[1] is used. The camera pose is computed to closely align the current oriented points with the previous frame by ICP. However, the depth data from the Kinect has some noise and ICP algorithm has limitations on alignment accuracy. Therefore, if we transfer 3D vertices in each frame of Kinect's camera coordinate into the world coordinate using the camera poses estimated by KinectFusion, there are some misalignments as shown in Fig. 1. This is the problem of KinectFusion and some papers mentioned this problem. For instance, Henry *et al.* uses both ICP and visual information for estimating more accurate camera pose[4]. Although many different approaches has been researched to solve this problem, the number of methods which use line segments on images are very few.

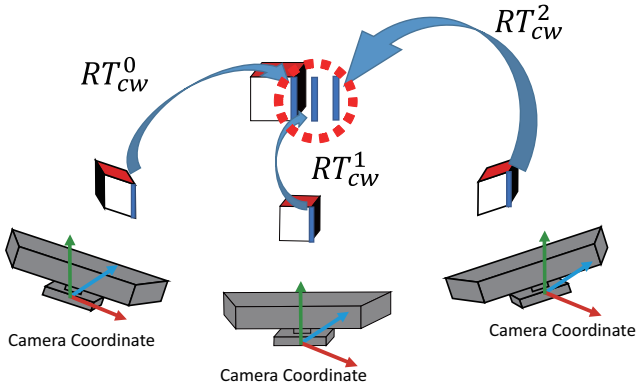


Fig. 1. KinectFusion's misalignment.

To solve this problem about point based alignment, we propose a method for alignment by using line segments. More accurate camera pose is estimated by this alignment. In this proposed method, 3D model is represented as line segment based model. Then 2D line segments in Kinect RGB images are matched with the 3D line segments in the 3D model to obtain 2D-3D line correspondence. We also propose an improvement of an existing method for line segment matching by LEHF[5].

The overall structure of this paper is organized as follows. Section II describes the accuracy of the camera pose obtained from KinectFusion and then Section III proposes an improvement of LEHF that is a line segment descriptor for 2D line segment matching. Section IV provides a detail of our proposed method for estimating the accurate camera poses. Experimental result and evaluation of alignment are in Section V and conclusion is discussed in Section VI.

The KinectFusion which we use in our system is the open-source *Kinfu* code in the Point Cloud Library (PCL) from Willow Garage[8].

II. The Accuracy of Camera Pose from KinectFusion

Using KinectFusion, we can obtain Kinect's RGB Image, Depth Image and the camera pose which translates from Kinect

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.374

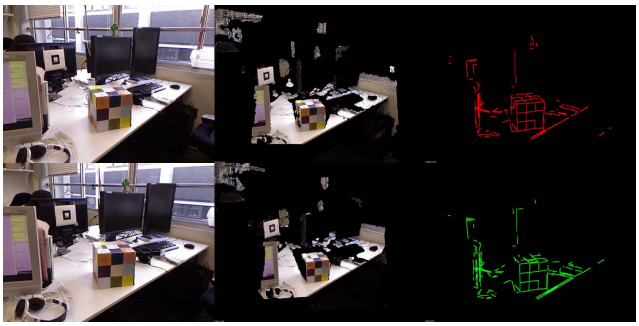


Fig. 2. 3D point based model and 3D line segment based model from 2 images.

Fig. 3. 3D point based model in the world coordinate.

camera coordinates to world coordinates in each frame. This camera pose at time i is a transform matrix $\mathbf{RT^i_{cw}} = [\mathbf{R_i}|\mathbf{t_i}]$ containing a 3×3 rotation matrix ($\mathbf{R_i}$) and 3D translation vector ($\mathbf{t_i}$). KinectFusion aligns the point cloud models using ICP algorithm and computes the camera poses. Suppose we have N RGB Images $\{I^N_{rgb}\}$, N Depth Images $\{I^N_d\}$ and N camera poses $\{RT^N_{cw}\}$ from KinectFusion, we can create 3D point based model and 3D line segment based model. Fig. 2 shows the two kinds of models from 2 RGB images on the left side. The models on the center of the figure are 3D point based and the right side ones are 3D line segment based. We explain how to create 3D line segment based model in Sec.IV. These 2 RGB images are obtained from KinecFusion, therefore we can also get the camera poses. If we transform the 3D models into the world coordinate, corresponded points and line segments are supposed to overlap, and Fig. 3 and Fig. 4 show the two 3D models in the world coordinate. The corresponded points and lines segments do not overlap. This result shows there are some errors in $\{RT^N_{cw}\}$. With that, our goal is to compute accurate camera poses, $\{correctedRT^N_{cw}\}$, by using line segments.

III. Improving LEHF

As we will explain in a later part of this paper, we use line segment matching for our proposed method. To properly match line segments, we adopt Line-based Eight-directional Histogram Feature (LEHF) which is a fast feature descriptor for line segments. Although LEHF is very efficient, it still poses one problem. LEHF has no concept of line segment direction. It does not define which edge point on the line segment is start point or end point. Therefore, LEHF is designed symmetrically and we need to compute two distances. One is between LEHFs that both of LEHF vectors are same

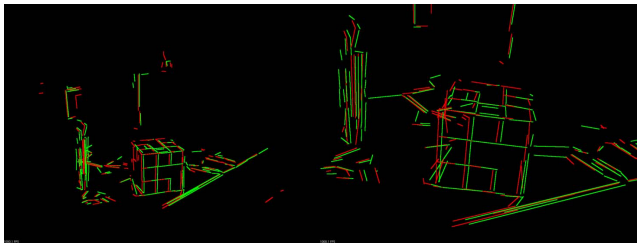


Fig. 4. 3D line segment based model in the world coordinate.

direction and another is between LEHFs that one LEHF vector is inverted. (See [5] for more details.)

To solve this problem, we add directional information to line segments. In other words, we choose start point and end point of line segments. To decide direction of a line segment is to decide the side of a line segment. Using the method discussed in [2], we define the side of a line. In this method, the side is defined by the image intensities gradient of line, which is defined as the average gradient of all the points on the line. The side which located on the region directed by the gradient of line is right side of the line, and the other is left side.

By using this method, we can define the right side and left side of line segments. Therefore, we can also define the start point and the end point of line segments as Fig. 5(a). Given the direction of line segment, we do not need to design LEHF symmetrically. Therefore we can define directed LEHF (Fig. 5(b)) and redefine $\mathbf{d}$ which is shown as eq.5 in [5] as

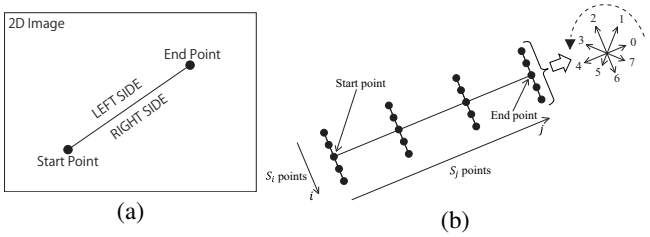
$$\begin{aligned}\mathbf{d} \quad = \quad & (h_{1,0}, \cdots, h_{1,7}, h_{2,0}, \cdots, h_{2,7}, h_{3,0}, \cdots, h_{3,7}, \\ & h_{4,0}, \cdots, h_{4,7}, h_{5,0}, \cdots, h_{5,7}), \end{aligned} \tag{1}$$

in which we assume that $S_i = 5$, 5 eight-directional gradient histograms ($\mathbf{h_i} = (h_{i,0}, \cdots, h_{i,7})$) are obtained because we do not count the center $\mathbf{h_i}$ twice. Thus, the distance need not to compute twice in LEHF matching. Hereafter, we use this directed LEHF.

Here, we demonstrate the performance of the directed LEHF by comparing with the performance of the original LEHF. We built a simple synthetic environment that shown in Fig. 6. Then 100 sequential images were generated with perfectly known camera poses. In this experiment, the camera pose was estimated by the tracking method explained in [5] and calculated errors between the estimated camera pose and ground truth in each case that we use the original LEHF and the directed LEHF. The error of translation vector from the camera pose was computed by Euclidean distance and the error of rotation matrix is computed by Eq. (2).

$$d(\mathbf{P}, \mathbf{Q}) = \sqrt{\frac{4\pi}{3} \sum_{i,j=1}^{3} |\mathbf{P_{ij}} - \mathbf{Q_{ij}}|^2}, \tag{2}$$

in which $\mathbf{P}$ is rotation matrix form the estimated camera pose and $\mathbf{Q}$ is ground truth. The experimental results are shown in Fig. 7 and Fig. 8. These result show the camera pose with the directed LEHF is more accurate than that with the original LEHF.



(a) (b)

Fig. 5. Improvment of LEHF. (a) The relation of edge points and side of a line segment, (b) Overview of directed LEHF.

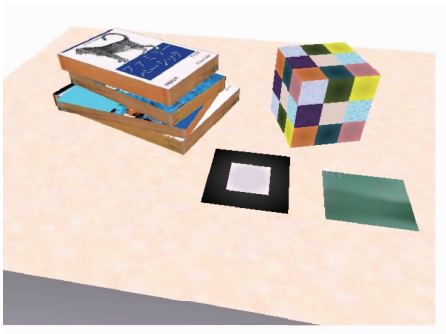


Fig. 6. Constructed synthetic environment.

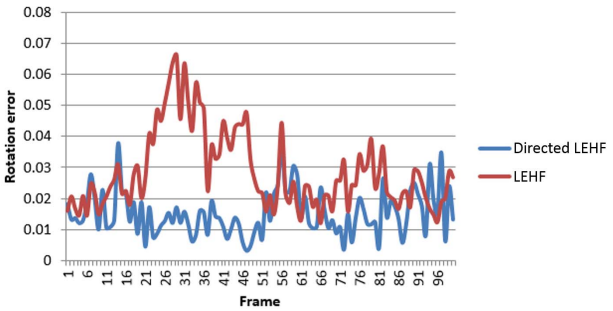


Fig. 7. The error of rotation matrix.

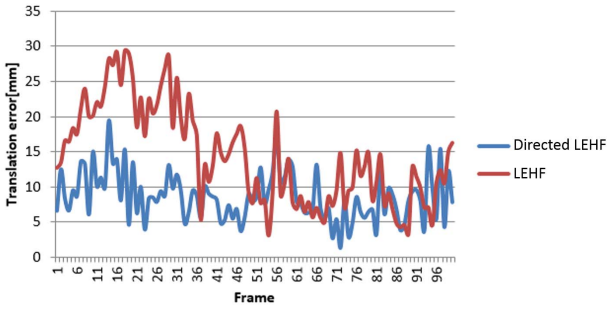


Fig. 8. The error of translation vector.

IV. Proposed Method

In this section, we explain the method to compute more accurate camera pose. Fig. 9 shows system overview of our proposed method. In this method, $\{I^N_{rgb}\}$, $\{I^N_d\}$ and $\{RT^N_{cw}\}$ are inputs and accurate camera poses, $\{correctedRT^N_{cw}\}$ are outputs. From top to bottom, we obtain a matching of 2D line segments in each RGB images with the directed LEHF. Then, we create 3D line segment based model and obtain correspondences between 2D and 3D line segments. From then on, using the 2D-3D line correspondences, we solve the Perspective-n-Lines (PnL) problem and finally obtain accurate camera poses. In the following subsection, we explain how to compute $correctedRT^i_{cw}(i \geq 1)$. Note that our proposed method uses RT^0_{cw} as the base because KinectFusion computes other camera poses based on the camera pose from first frame. KinectFusion decides that the 3D translation vector of the first frame camera pose ($\mathbf{t_0}$) is $(1.5, 1.5, -0.3)^t$. Therefore, we can define $correctedRT^0_{cw}$ as

$$correctedRT^0_{cw} = RT^0_{cw} = \begin{bmatrix} 1 & 0 & 0 & 1.5 \\ 0 & 1 & 0 & 1.5 \\ 0 & 0 & 1 & -0.3 \end{bmatrix}. \quad (3)$$

A. Obtain 2D line segment matching

First, 2D line segments are extracted from RGB images. In this line segment extraction, a fast line segment detector (LSD)[11] is applied. Using LSD, we obtain edge points of 2D line segment. Given two sets of line segments extracted from I^{i-1}_{rgb} and I^i_{rgb} by LSD, $\mathcal{L}^{i-1} = \{l^1_{i-1}, l^2_{i-1}, \cdots, l^{M_{i-1}}_{i-1}\}$ and $\mathcal{L}^i = \{l^1_i, l^2_i, \cdots, l^{M_i}_i\}$, respectively, we search for 2D line segment matching with $\mathcal{L}^{i-1}$ and $\mathcal{L}^i$ by using the directed LEHF matching. The resulting set of matching 2D line segments is represented as

$$\mathcal{LM}^i = \{(l^{g(j)}_{i-1}, l^{f(j)}_i), j = 1, 2, \cdots, N^i\}, \quad (4)$$

in which $(l^{g(j)}_{i-1}, l^{f(j)}_i)$ represents a pair of matching 2D lines and $g(j) \in [1, M_{i-1}]$, $f(j) \in [1, M_i]$. (Fig. 10)

B. Create 3D line segment based model

2D line segment's start point and end point are translated from Image coordinates to Kinect camera coordinates using Depth Image. Therefore, connecting these two 3D edge points, we obtain a 3D line segment. Each line segment in $\mathcal{L}^{i-1}$ and $\mathcal{L}^i$ is transformed to 3D line segment by this procedure and we obtain $A^{i-1} = \{L^1_{i-1}, L^2_{i-1}, \cdots, L^{M_{i-1}}_{i-1}\}$ and $A^i = \{L^1_i, L^2_i, \cdots, L^{M_i}_i\}$. A^{i-1} and A^i are 3D line segment based models in each of the Kinect camera coordinate. Then A^{i-1} and A^i are translated into the world coordinates by RT^{i-1}_{cw} and RT^i_{cw}. Therefore we can get $\mathcal{L}^{i-1}_{3D} = \{L^1_{w,i-1}, L^2_{w,i-1}, \cdots, L^{M_{i-1}}_{w,i-1}\}$ and $\mathcal{L}^i_{3D} = \{L^1_{w,i}, L^2_{w,i}, \cdots, L^{M_i}_{w,i}\}$, where $L_{w,i}$ is translated from L_i into the world coordinate by RT^i_{cw}. Because we have 2D line segment matching, $\mathcal{LM}^i$ from $\mathcal{L}^{i-1}$ and $\mathcal{L}^i$, we can also obtain 3D line segment matching from $\mathcal{L}^{i-1}_{3D}$ and $\mathcal{L}^i_{3D}$. The set of matching 3D line segments is represented as

$$\mathcal{LM}^i_{3D} = \{(L^{g(j)}_{w,i-1}, L^{f(j)}_{w,i}), j = 1, 2, \cdots, N^i\}, \quad (5)$$

in which $(L^{g(j)}_{w,i-1}, L^{f(j)}_{w,i})$ represents a pair of matching 3D lines and $g(j) \in [1, M_{i-1}]$, $f(j) \in [1, M_i]$.(Fig. 11)

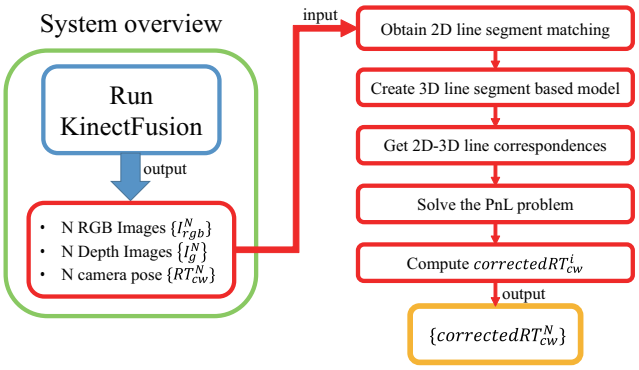


Fig. 9. System overview of our proposed method.

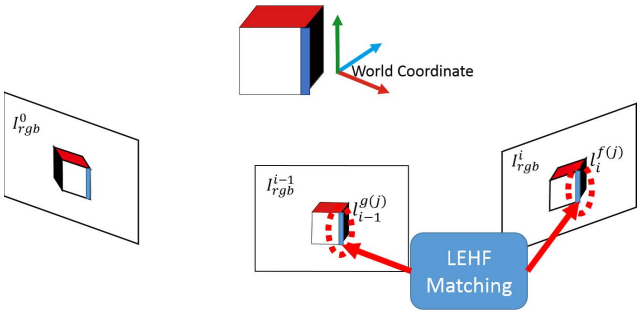


Fig. 10. 2D line segment matching by LEHF.

C. Get 2D-3D line correspondences

In an ideal situation, $L^{g(j)}_{w,i-1}$ and $L^{f(j)}_{w,i}$ are supposed to overlap each other in the world coordinates. However, as we saw in Fig. 4, they do not overlap because of the error in RT^i_{cw}. 2D lines from $\mathcal{L}^i$ and 3D lines from $\mathcal{L}^{i-1}_{3D}$ brought to correspondence to align $\mathcal{L}^i_{3D}$ with $\mathcal{L}^{i-1}_{3D}$, so that the camera pose can be computed from the 2D-3D line correspondences. The set of 2D-3D line correspondences is represented as

$$\mathcal{LC}^i = \{(L^{g(j)}_{w,i-1}, l^{f(j)}_i), j = 1, 2, \cdots, N^i\}, \quad (6)$$

in which $(L^{g(j)}_{w,i-1}, l^{f(j)}_i)$ represents a pair of 2D-3D line correspondences and $g(j) \in [1, M_{i-1}]$, $f(j) \in [1, M_i]$.(Fig. 12)

D. solve the PnL problem

If we had 2D-3D point correspondences, we would solve the PnP problem and obtain the camera pose. However, this method uses line segments, therefore we must solve not the PnP problem but the PnL problem. Given a set of 2D-3D line correspondences, we can solve the PnL problem to recompute the camera pose by using RPnL[12]. In practice, because there might be mismatches of line correspondences, we show how to solve the PnL problem with an algorithm like RANSAC[3].

Suppose we have $\mathcal{LC}^i$ which is N^i sets of 2D-3D line correspondences, we randomly select four 2D-3D line correspondences from $\mathcal{LC}^i$ because the program of RPnL needs at least four correspondences. Let the four set of 2D-3D line correspondences be represented as

$$\mathcal{LC}^i_{four} = \{(L^{a(k)}_{w,i-1}, l^{b(k)}_i), k = 1, 2, 3, 4\}, \quad (7)$$

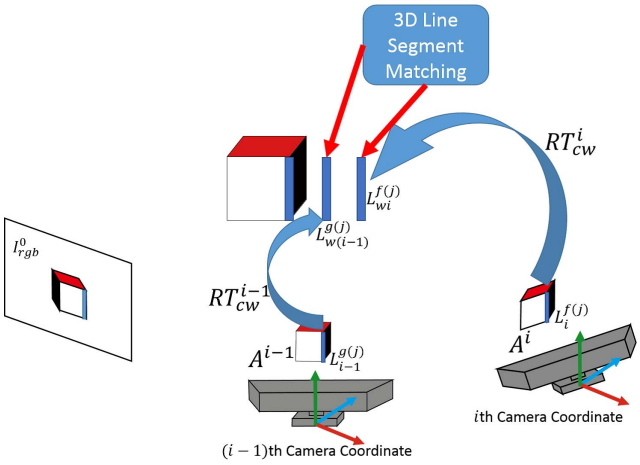


Fig. 11. 3D line segment matching

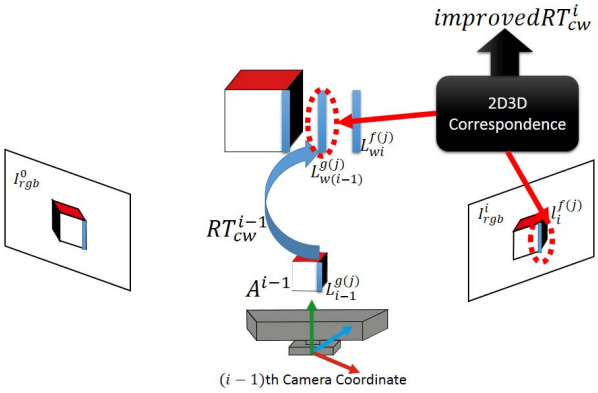


Fig. 12. 2D-3D line correspondences.

in which $(L^{a(k)}_{w,i-1}, l^{b(k)}_i)$ represents four pairs of 2D-3D line correspondences and $a(k) \in [1, M_{i-1}]$, $b(k) \in [1, M_i]$. Moreover, the rest of $(N^i - 4)$ 2D-3D line correspondences are represented as

$$\mathcal{LC}^i_{rest} = \{(L^{g(j)}_{w,i-1}, l^{f(j)}_i) | 1 \le j \le N^i, g(j) \neq a(k), f(j) \neq b(k), k = 1, 2, 3, 4\}. \quad (8)$$

With $\mathcal{LC}^i_{four}$, we solve the PnL problem using RPnL and estimate the camera pose. Then, each 2D line segment, $l^{f(j)}_i$ from $\mathcal{LC}^i_{rest}$, is translated to 3D line segment in the world coordinates, $L'^{f(j)}_i$, by using depth value and estimated camera pose. We calculate the error, $e(j)$, between $L^{g(j)}_{w,i-1}$ from $\mathcal{LC}^i_{rest}$ and $L'^{f(j)}_i$. Let $e(j)$ be $e(j) = S(j)/(length_{w,i-1} + length_i)$, where $S(j)$ is an area of rectangle obtained by connecting four edge points of $L^{g(j)}_{w,i-1}$ and $L'^{f(j)}_i$, $length_{w,i-1}$ is length of $L^{g(j)}_{w,i-1}$, and $length_i$ is length of $L'^{f(j)}_i$. The total of $e(j)$ is defined as error of the estimated camera pose and its unit is pixel.

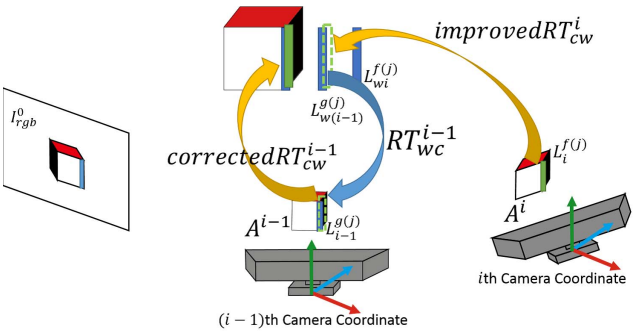


Fig. 13. compute $correctedRT^i_{cw}$.

We also randomly select another four sets of $\mathcal{LC}^i_{four}$ and repeat the steps explained above N_{RANSAC} times to estimate the camera pose. We choose the estimated camera pose which has the smallest total of $e(j)$ as a tentative camera pose. Next, using this tentative camera pose and depth value, we translate each 2D line segment, $l^{f(j)}_i$ to 3D line segment in the world coordinates, $L'^{f(j)}_i$. We calculate $e(j)$ and if $e(j)$ is less than threshold (TH_e), we save the 2D-3D line segment correspondence as inlier.

Finally, we compute the camera pose which aligns $\mathcal{L}^i_{3D}$ with $\mathcal{L}^{i-1}_{3D}$ by another algorithm for the PnL problem proposed by Kumar and Hnson[6]. This algorithm estimates the camera pose iteratively and needs a set of 2D-3D line segment correspondence and initial camera pose as inputs. We take the inlier and the tentative camera pose as inputs and obtain an improved camera pose, $improvedRT^i_{cw}$ as output of the algorithm.

The procedure discussed above solves the PnL problem with a RANSAC[3] like algorithm and this solution gives us $improvedRT^i_{cw}$. Note that $improvedRT^i_{cw}$ is a camera pose that translates A^i from the Kinect camera coordinate to A^i_w in the world coordinates and A^i_w is aligned to $\mathcal{L}^{i-1}_{3D}$.

E. compute $correctedRT^i_{cw}$

After computing $improvedRT^i_{cw}$, we can obtain A^i_w. Then we translate A^i_w from the world coordinate to (i-1)th Kinect camera coordinate by RT^{i-1}_{wc}, where $RT^{i-1}_{wc} = (RT^{i-1}_{cw})^{-1}$. Lastly, this translated A^i_w in the (i-1)th Kinect camera coordinates is translated into the world coordinates by $correctedRT^{i-1}_{cw}$ and aligned to the 3D line segment based model of the first frame. (Fig. 13) To summarize these steps, we can represent $correctedRT^i_{cw}$ as

$$correctedRT^i_{cw} = correctedRT^{i-1}_{cw} \cdot RT^{i-1}_{wc} \cdot improvedRT^i_{cw} (i \geq 1). \quad (9)$$

Therefore, we can obtain accurate 3D line segment models in the world coordinates by using $correctedRT^i_{cw}$ instead of RT^i_{cw}. Fig. 14 shows the result of $correctedRT^i_{cw}$'s translation of the two 3D line segment based model from Fig. 2. Compared with Fig. 4, the two models in Fig. 14 overlap.

V. Evaluation

In this section, we try to evaluate the accuracy of the camera poses which are computed by our proposed method.

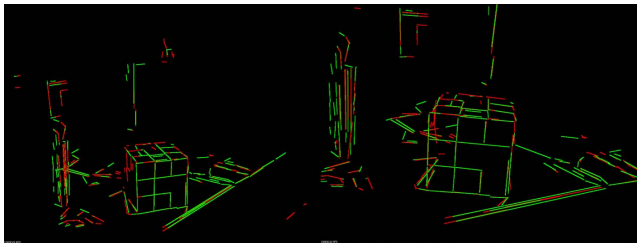
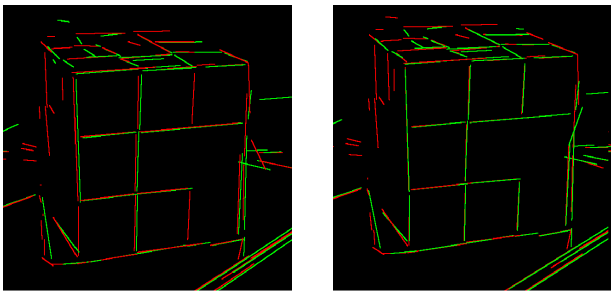


Fig. 14. Aligned 3D line segment based model from 2 images using our proposed method.



(a) (b)

Fig. 15. The alignment result of the two 3D line segment based model. (a) using ICP, (b) using our method.

To evaluate it, we experiment creating 3D line segment based model. In this experiment, we set N_{RANSAC} to 2000 and TH_e to 0.003.

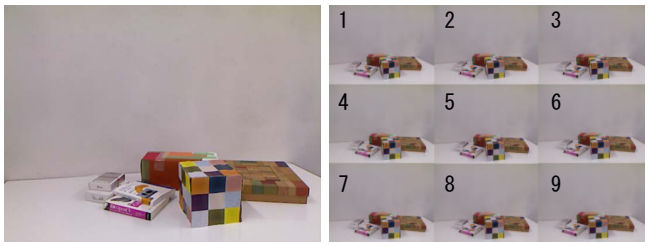
A. Comparison of ICP and proposed method

Suppose we had two 3D line segment based models in world coordinates as shown in Fig. 4, we tried to align the two models by using ICP and our proposed method. We used points on the line segments of models for ICP. Fig. 15 shows the results of alignment. Fig. 15(a) is an alignment result by ICP and (b) is by our proposed method. Our method aligned the models more accurately.

B. The accuracy of object shape reconstruction

Moreover, we evaluate the accuracy of object shape reconstruction using the camera pose estimated by our method. Fig. 16 shows 10 RGB Images from KinectFusion. Fig. 16(a) is the base image of alignment and Fig. 16(b) shows other 9 RGB Images. We used these 10 RGB Images, Depth Images and camera poses which KinectFusion estimated as Inputs for our system, and estimated the accurate camera poses. Using these camera poses, we reconstructed object shape. Fig. 17 and Fig. 18 show the result of reconstruction. Fig. 17 is the virtual view images generated by the camera poses which KinectFusion estimated, and Fig. 18 is the one generated by the camera pose estimated by our proposed method.This result shows the virtual view images from our proposed method are more accurate than the one from KinectFusion because the parts of object overlap perfectly. Next, we represented the object shape as point cloud model in each frame and compute the distances between points from the base frame's model and points from the ith frame's model. ($1 \leq i \leq 9$) The result

of this evaluation is showed in Fig. 19. This graph shows that the distances between points from models reconstructed by our method are smaller than the distances between points from models reconstructed by KinectFusion. This means our method estimated more accurate camera pose and obtained more accurate object shape.



(a) (b)

Fig. 16. Input images (a) the base image, (b) other images.

Fig. 17. Result of object shape reconstruction by KinectFusion.

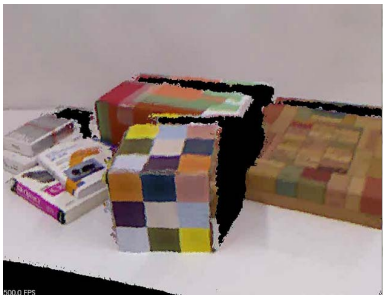


Fig. 18. Result of object shape reconstruction by proposed method.

VI. Conclusion

KinectFusion estimates the camera pose by ICP algorithm in real time. However, the estimated camera pose lack in accuracy. Although a number of approach has been done to solve this problem, very few methods use line segment matching for estimating the camera pose. In this paper, we propose a method for improving the accuracy of camera pose estimation for KinectFusion by using line segment correspondents. We also propose an improve version of the line matching method, LEHF. Our proposed method firstly obtains 2D line matching and then create 3D line segments. Next, 2D-3D line segment correspondents are obtained and we estimate the accurate

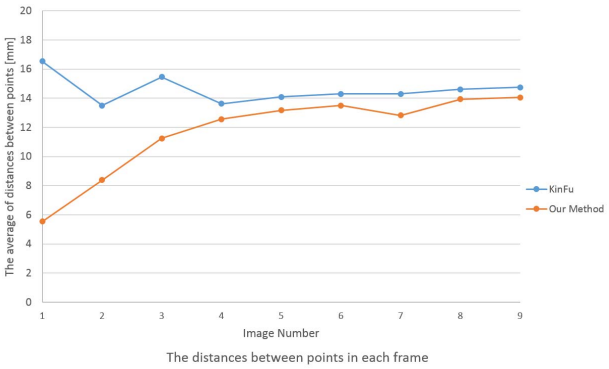


Fig. 19. The distances between points in each frame.

camera pose by solving the PnL problem with an algorithm like RANSAC. We showed that the estimated camera poses have more accuracy than the original ones.

References

[1] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[2] Bin Fan, Fuchao Wu, and Zhanyi Hu. Line matching leveraged by point correspondences. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 390–397. IEEE, 2010.

[3] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[4] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25, 2010.

[5] Keisuke Hirose and Hideo Saito. Fast line description for line-based slam. In *BMVC*, pages 1–11, 2012.

[6] Rakesh Kumar and Allen R Hanson. Robust methods for estimating pose and a sensitivity analysis. *CVGIP: Image Understanding*, 60(3):313–342, 1994.

[7] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[8] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[9] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 519–528. IEEE, 2006.

[10] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[11] R Grompone Von Gioi, Jeremie Jakubowicz, J-M Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(4):722–732, 2010.

[12] Lilian Zhang, Chi Xu, Kok-Meng Lee, and Reinhard Koch. Robust and efficient pose estimation from line correspondences. In *Computer Vision–ACCV 2012*, pages 217–230. Springer, 2013.