A Localization Method for Markerless AR Using a Depth Image Sensor

Kenji Yamakawa
School of Engineering
Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Aichi, Japan

Katsuhiko Kaji and Nobuo Kawaguchi
Graduate School of Engineering
Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Aichi, Japan

***Abstract*—Our final goal is to realize an AR system for indoor environments, especially for visualizing power usage of electric appliances. We call it Power Scope. To show the desired information over the corresponding objects, it is necessary to identify objects. To achieve it, we use a depth image sensor and perform two kinds of localization method, approximate localization and accurate localization. The former is a method to estimate where the current room is. The latter is a method to estimate where the scope is located in the room. In this paper, we mainly introduce the former and exemplified the usefulness of it. For approximate localization, we prepare the feature models of the rooms beforehand. Features are calculated based on the planar objects in rooms. And then the system identifies the current room by comparing the current state with them.**

***Index Terms*—Point Cloud, Planar Objects, Markerless AR.**

I. Introduction

Environmental problems are regarded as important these days. One of them is the electrical energy. It is important to save electricity at home. Smart house which generates, stores and controls energy has come into the world limelight. One of its functions is to visualize electricity usage. It encourages users to reduce electricity consumption. They can see the visualized information with smart phones, computers and so on[1]. In the current system, visualized information mainly consists of numerical values. It is hard for users to imagine the status of the energy.

Our research aims at realizing an augmented reality (AR) system for smart houses in order to solve such problems. AR is a technology to augment the real-world environment by computer-generated images. We are planning to visualize power usage of electric appliances in our system. We call it *Power Scope*. Using the scope, users check the electricity usage intuitively. The contents of AR are considered as follows: Status of electricity consumption is displayed over each electricity appliance, status of electricity distribution over basic components of rooms such as walls, ceilings and floors. The contents are not only statistical data but also animations and personified agents. Therefore users are able to understand the whole electricity conditions of their house intuitively, which makes them environmentally aware.

In order to realize such AR system, we propose two kinds of localization method. One is a method to grasp where the current room is. We call it *approximate localization* in this paper. The other is a method to grasp where the scope is located in the room. We call this step *accurate localization* in this paper. There are some rooms in a house, and there are a lot of objects in a room. Some kinds of objects may be not only in one room but also in the other rooms such as desks or TVs. To identify such same objects, it is required to perform approximate localization. We prepare the feature models of some specific environments, namely targets of AR, in the rooms beforehand. Assuming that the object placements in a room are different from the other room, we identify the current room by comparing the current status with the models. After that, it is required to perform accurate localization in order to identify objects. We estimate a relationship between the scope coordinate and the real world coordinate to find the corresponding objects. Object identification is required in order to show the information over the desired objects.

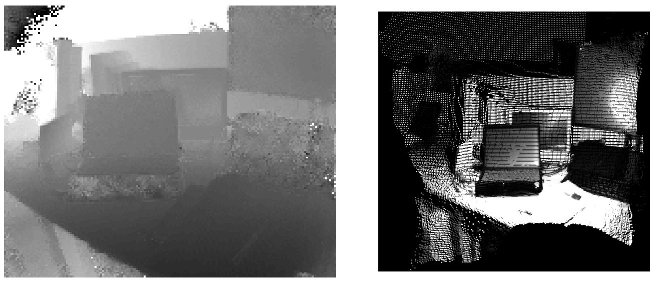
Fig. 1. Depth Image (left) and Point Cloud (right)

We acquire the real environment data by using a depth image sensor. A depth image contains information relating to the distance of the surfaces of objects from a viewpoint of sensors. Figure 1 shows an example of it. The darker area means near. A depth image can be converted into a point cloud. A point cloud is a set of discrete points distributed in the 3D space. In this research we use Mesa Imaging SwissRanger SR4000, $5m$ detection range and Wide field of view version.

In the followings, we first introduce the related works based on sensor types, then we propose our localization method. At the last, we exemplify the usefulness and weakness of our method by the experiment.

II. Related Work

Currently three kinds of sensors are mainly used to estimate the position and the direction in the real world coordinate system.

A. Positioning Sensor

Positioning sensors such as GPS sensors, magnetometric sensors and accelerometers provide a longitude and a latitude, a direction and a tilt of the sensor respectively. Based on them, we can estimate the direction and position. Smart phones are equipped with such sensors and are well matched with this kind of AR. It is generally used for large-scale environments, for example displaying what kind of facilities locates in the direction which the user is currently looking at.

B. Camera

There are two kinds of methods to realize AR with cameras. One is a method to use special markers, called AR markers. A position and a direction are estimated in real time by recognizing transformations and patterns of markers in an image. Kato developed ARToolKit[2]. It utilizes black square markers with unique pattern in it. It calculates the pose and the position of the marker detected in an image. Then virtual objects can be drawn in the marker coordinates.

There is the other approach which doesn't require such markers, called *markerless AR*. It uses interest keypoints in an image instead of such markers. Klein proposed Parallel Tracking and Mapping (PTAM)[3]. It is one of the major camera tracking techniques and available in AR. It extracts a lot of keypoints from images and reliable ones, namely observed ones many times, are used for estimating a position and a direction of the camera.

PTAM doesn't have the function to show specific AR contents on specific targets. Castle proposed Parallel Tracking and Multiple Mapping (PTAMM)[4] to add multiple map support to PTAM. PTAMM learns small spaces as different maps and finds the most similar map to the current map in real time. Thus PTAMM improves the scalability of PTAM.

C. Depth Image Sensor

A depth image sensor provides the discrete 3D information in the real world. Though there are some approaches to measure the distance between the surface of objects and the sensor, most of those are common in using infrared light. A depth sensor emit the infrared light to the target and calculate the distance by some methods.

Izadi presented KinectFusion[5] for real-time detailed 3D reconstructions of indoor scenes using only the depth data from a RGB-D sensor, Microsoft Kinect. The depth data is converted from image coordinates into 3D points and normals in the coordinate space of the sensor, and then a rigid 6DOF transform is computed from the current oriented points and the previous frame on a GPU. KinectFusion also realizes geometry-aware AR and physics-based interactions by using a real-time 3D model.

Rusu introduced a mapping system[6] that acquires 3D object models of man-made indoor environments for autonomous household robots. The system classifies a variety of objects from point clouds on the basis of functional reasoning. There are a lot of planar objects inside rooms, for example tables, floors, doors, drawers, shelves, etc. Firstly, it segments planes into two kinds, horizontal or vertical against the ground. Secondly, they are categorized into the real objects by its size, existence of fixed objects such as knobs and handles, and so on. Finally, robots act on the basis of it.

Positioning sensors are not suited for small-scale environments, such as rooms. It is because that errors become too large to ignore. Cameras are under the influence of the ambient light. But depth image sensors are robust against the great changes in the ambient light, such as morning and evening. Thus we decided to use a depth image sensor.

In Rusu's research, it is easy to estimate its rough position and direction since the depth image sensor is fixed on the robot. Furthermore they assume that enough point clouds are obtained. But in our research, users move the sensor freely in the 3D space. Furthermore we must interpret the current circumstances from one point cloud.

III. Localization Method

In this section, we propose a method to localize using a depth image sensor. For the multiple target AR, it is necessary to make estimate of the current location where the depth image was taken. We represent an environment as a point cloud. A point cloud can be made from a depth image.

The general outline of our method is depicted in Figure 2. Firstly, we take some point clouds of specific environments, say *scenes*. Then we compute features and store them into a database beforehand. Secondly, we roughly localize by comparing the features of the current state with the models in the database. The most closest models are found and provide the approximate estimate about the current position and direction. Finally, an accurate relationship between the sensor coordinate and the real world coordinate is calculated.

Features of scenes are based on planar objects existing in an obtained environment data. In indoor environments there are many planar objects, for example tables, floors, walls and electricity appliances as shown in Figure 3. Generally, large planes tend to stay where they are even as time goes by. There are three phases for the calculation of the scene features: First phase is to reduce noise in an obtained point cloud. Second phase is to extract reliable planes from the point cloud. Last phase is to compute the features from the planes.

A. Scene Features Representation

1) Noise Reduction: Depth images from a sensor contain noise. First of all we need to reduce it. For each point a k-nearest neighbor search is performed and distances to k points are computed. If one of them exceeds a threshold d_1, the point is added to the rejection list. After searching all points, the points in the list are removed. Figure 4 shows the result of the noise reduction. Complicated shapes are lost, but it doesn't matter for plane extraction. In this paper, we set $k = 20$ from the experimental knowledge.

2) Plane Extraction: This step is mainly based on Rusu's methods[6]. Firstly, surface normals are estimated. In this paper, 20 nearest points are selected and then a normal of each point is computed. And then a plane segmentation is

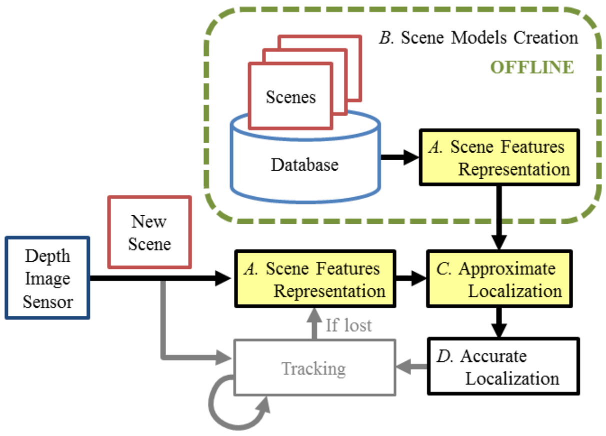

Fig. 2. Outline of Our Method

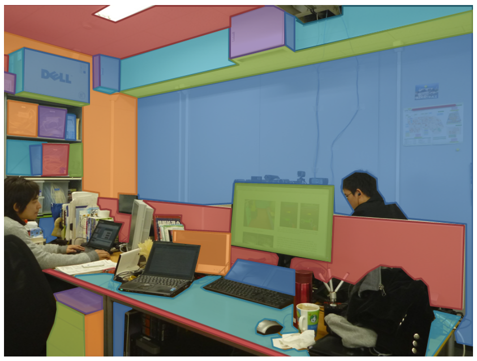

Fig. 3. Planar Objects

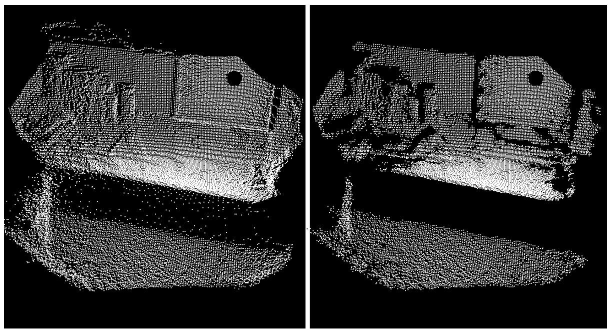

Fig. 4. Raw Point Cloud (left) and Result of Noise Reduction (right)

performed. Three points in a point cloud $\mathcal{P}$ are randomly selected to create an equation of a plane P in 3D space (x, y, z):

$$\alpha_P x + \beta_P y + \gamma_P z + \delta_P = 0 \tag{1}$$

where $\alpha_P, \beta_P, \gamma_P, \delta_P$ are in a Hessian normal form. To be more precise, a normal vector $\boldsymbol{n}_P$ of a plane P is defined as:

$$\boldsymbol{n}_P = (\alpha_P, \beta_P, \gamma_P) \tag{2}$$

$$\|\boldsymbol{n}_P\| = 1 \tag{3}$$

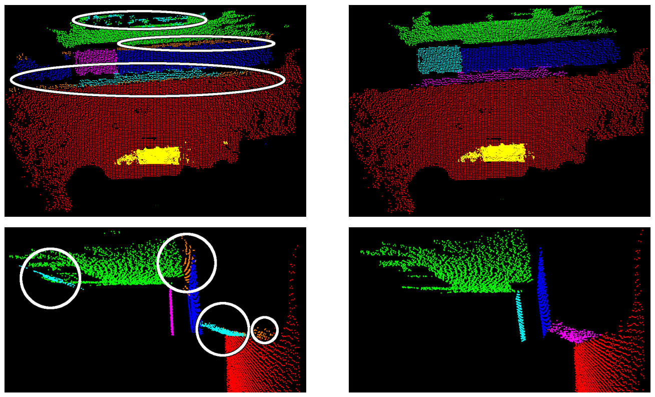

Fig. 5. Effect of Clustering: Not performed (left column) and performed (right column), View from front (top row) and right (bottom row). The light blue and orange planes are detected in error.

and a distance d between a point $\boldsymbol{p}$ and P can be calculated by:

$$d = \boldsymbol{n}_P \cdot \boldsymbol{p} + \delta_P \tag{4}$$

The number of points within a threshold d_2 from the plane is counted. By repeating this steps, the largest plane, that is, the plane which contains the most points is extracted from the point cloud. This floating points are projected onto the plane.

Segmented planes have outliers. To get rid of them, we consider surface normals. Angles between the normals of the estimated plane and of the points are calculated. Those of them which become more than a threshold θ_s are weeded out.

After that, the largest cluster is extracted from the plane. For every point, a set of point neighbors in a sphere with radius d_3 is searched for and grouped. Figure 5 presents the difference whether the clustering process is performed or not. Same colors mean same planes. If the clustering was not performed, wrong planes may be detected as surrounded by white frames in the left column of Figure 5.

The above-mentioned original methods by Rusu[6] use the fixed thresholds d_1, d_2 and d_3, assuming a constant point density throughout a point cloud. But in our research, an input point cloud is sparse since they are created from only one depth image. The density partially vacillates due to the depth from the viewpoint. That is to say, the closer to the sensor, the more dense. Therefore we set $d_i (i = 1, 2, 3)$ as a dynamic number defined as follows. Two points are assigned in a same group if their gap is smaller than:

$$d_i = \delta_i p_z \tag{5}$$

where δ_i is a parameter. We use $\delta_1 = 0.03m$, $\delta_2 = 0.02m$, $\delta_3 = 0.05m$ in this paper. The thresholds changes in proportion to the depth p_z. In the same way, it is also better for θ_s to be set a dynamic value rather than a static one. The further the points are from the viewpoint, the larger measurement errors are. It results in larger errors of surface normals far from the viewpoint. We defined θ_s as:

$$\theta_s = \phi\sqrt{p_z} \tag{6}$$

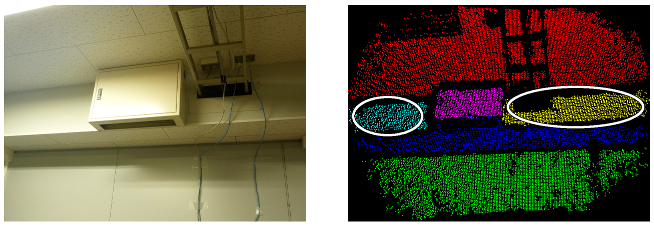

Fig. 6. Recognition as Different Planes: The real environment (left) and the extracted planes (right). The light blue and yellow planes should be detected as the same plane.

where ϕ is a parameter. In this paper, we set $\phi = 10^\circ$ so that the angle threshold should be 10° in the near space, $1m$ distant from the viewpoint, and about 20° in the far space, $4m$ distant.

By repeating the above steps, m planes are extracted. The iteration is discontinued if the number of points in a extracted plane is under t. Small planes are not reliable in this method. There are two reasons for it. One is that they tend to move over time such as books. The other is that the segmentation algorithm works properly for sufficient inliers. Assuming the sensor keeps more than $1m$ distant from the target, $t = 150$ is used in this paper. Therefore the areas of the detected planes are more than about $400cm^2$.

The actual one plane may be recognized as different planes because of the existence of some object in front of them as shown in Figure 6. It is also caused by the clustering process introduced above. But if it was not done, incorrect planes would be extracted as shown in Figure 5. To solve this problem, a plane integration process is required. A plane P_1 is combined with P_2 if:

$$\boldsymbol{n}_{P_1} \cdot \boldsymbol{n}_{P_2} > \cos\theta_{\text{th}} \tag{7}$$

$$\boldsymbol{p}_{\text{center}_1} \cdot \boldsymbol{c}_{P_2} < d_{\text{th}} \tag{8}$$

$$\boldsymbol{p}_{\text{center}_2} \cdot \boldsymbol{c}_{P_1} < d_{\text{th}} \tag{9}$$

where $\boldsymbol{n}_{P_1}$ and $\boldsymbol{n}_{P_2}$ are the normals of P_1 and P_2. A threshold θ_{th} is set as $\theta_{\text{th}} = 20^\circ$ in this paper. The points $\boldsymbol{p}_{\text{center}_1}$ and $\boldsymbol{p}_{\text{center}_2}$ are the centroids of P_1 and P_2. The vectors $\boldsymbol{c}_{P_1}$ and $\boldsymbol{c}_{P_2}$ are made from the coefficients of the equations of P_1 and P_2 defined as:

$$\boldsymbol{c}_{P_i} = (\alpha_{P_i}, \beta_{P_i}, \gamma_{P_i}, \delta_{P_i}) \quad (i = 1, 2) \tag{10}$$

where $\alpha_{P_i}, \beta_{P_i}, \gamma_{P_i}, \delta_{P_i}$ are the coefficients described in Equation 1 Therefore the left sides of Inequality 8 and 9 represent the distances between $\boldsymbol{p}_{\text{center}_1}$ and P_2, and $\boldsymbol{p}_{\text{center}_2}$ and P_1. In this paper, we use $d_{\text{th}} = 0.05m$. After applying this step to all the combinations of the m planes, the number of planes decreases to n. Figure 7 shows a good result of plane extraction.

3) Features Calculation: We define the scene features as follows:

- The number of the planes n
- The angles between all the planes $\boldsymbol{A}$
- The distances between all the parallel planes $\boldsymbol{D}$

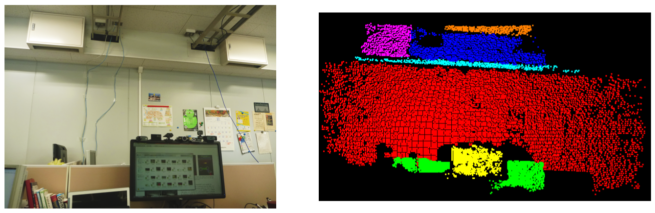

Fig. 7. Successful Result of Plane Extraction: The real environment (left) and the extracted planes (right)

The number of the obtained planes n changes according to the viewpoint. Therefore it can be used to narrow the candidates roughly.

A relationship of angles between planes doesn't depend on a position and a tilt of the sensor, moreover areas of planes and time. Therefore, for all the ${}_nC_2$ combinations an angle $a_{i,j}$ between the planes P_i and P_j $(i, j = 1, 2, \cdots, n; i < j)$ is calculated and stored as a component of an upper triangular matrix:

$$\boldsymbol{A} = \begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ & 0 & \ddots & \vdots \\ & & 0 & a_{n-1,n} \\ O & & & 0 \end{pmatrix} \tag{11}$$

In the real world, there are a lot of parallel and perpendicular planes. It is expected that the most angles are close to either 0° or 90°. In this case, it is difficult to match the scenes correctly. Thus we utilize distances between parallel planes. We regard as parallel if an angle between two planes are under 20° in this paper. A distance $d_{i,j}$ between the planes P_i and P_j $(i, j = 1, 2, \cdots, n)$ is computed from the forth coefficients δ_P in Equation 1:

$$d_{i,j} = \begin{cases} |\delta_{P_i} - \delta_{P_j}| & \text{(if parallel)} \\ 0 & \text{(otherwise)} \end{cases} \tag{12}$$

According to the Hessian normal form, δ_{P_1} and δ_{P_2} are equal to the distances between the planes and the origin. If two planes are parallel, the difference of δ is nearly equal to the distance between them. If they are not parallel, $d_{i,j}$ is set as a constant zero. These are stored as components of an upper triangular matrix:

$$\boldsymbol{D} = \begin{pmatrix} 0 & d_{1,2} & \cdots & d_{1,n} \\ & 0 & \ddots & \vdots \\ & & 0 & d_{n-1,n} \\ O & & & 0 \end{pmatrix} \tag{13}$$

These matrices $\boldsymbol{A}$ and $\boldsymbol{D}$ are utilized as lookup tables in the following phase.

B. Scene Models Creation

Scenes change according to pose of the sensor. Therefore for each target environment, many scenes from different viewpoints are taken. By the methods described in Section III-A, n acquired scenes are processed. Generated models $m_i (i = 1, 2, \cdots, n)$ are stored into a database M. The structure of a model m_i is depicted in Figure 8.

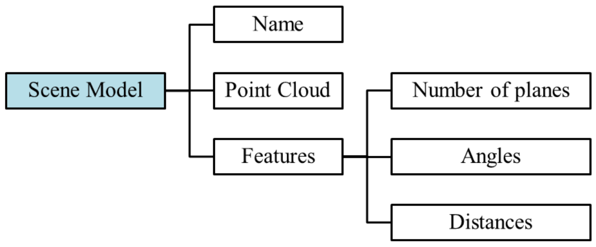

Fig. 8. Model Structure

TABLE I
FEATURE MATCHING ALGORITHM

```
Algorithm Feature_Matching$(m_i, m_{\mathrm{curr}})$ :
1:  $a_{\mathrm{best}} := a_{\mathrm{th}}, p_{\mathrm{best}} := null$
2:  if neither $m_i$ nor $m_{\mathrm{curr}}$ include any parallel planes do
3:    $d_{\mathrm{best}} := 0$
4:  else
5:    $d_{\mathrm{best}} := d_{\mathrm{th}}$
6:  end if
7:  if $m_i$ doesn't have the same number of planes $n$ as $m_{\mathrm{curr}}$
8:    return $null$
9:  end if
10:  create permutations $p_n^k$ $(k = 1, 2, \cdot, n!)$
11:  for $k = 1$ to $n!$ do
12:    $a_{\max} := \max \left\{ |s_{u,v}| \middle| \boldsymbol{S} = [s_{u,v}] = \boldsymbol{A}_i - \boldsymbol{A}_{\mathrm{curr}}^{p_n^k} \right\}$
13:    $d_{\max} := \max \left\{ |t_{u,v}| \middle| \boldsymbol{T} = [t_{u,v}] = \boldsymbol{D}_i - \boldsymbol{D}_{\mathrm{curr}}^{p_n^k} \right\}$
14:    if $a_{\mathrm{best}} >= a_{\max} \cap d_{\mathrm{best}} >= d_{\max}$ do
15:      $p_{\mathrm{best}} := p_n^k$
16:      $a_{\mathrm{best}} := a_{\max}$
17:        $d_{\mathrm{best}} := d_{\max}$
18:      end if
19:    end if
20:  end for
21:  return $(p_{\mathrm{best}}, a_{\mathrm{best}}, d_{\mathrm{best}})$
```

C. Approximate Localization (Feature Matching)

A current scene is processed by the methods described above, and an input model m_{curr} is created. We compare m_{curr} with every model m_i in the database M as outlined in Table I. The thresholds a_{th} and d_{th} are set as $a_{\mathrm{th}} = 20°$ and $d_{\mathrm{th}} = 0.3m$ in this paper. If neither m_{curr} nor m_i include any parallel planes, d_{best} is set as $d_{\mathrm{best}} = 0$ in order to disable the feature. The number k is an index number of permutations from 1 to $n!$. The matrices $\boldsymbol{A}_{\mathrm{curr}}^{p_n^k}$ and $\boldsymbol{D}_{\mathrm{curr}}^{p_n^k}$ described in Line 12 and 13 mean that the coefficients of matrices are aligned respectively according to the permutation p_n^k. The returned a_{best} and d_{best} can be used as matching scores. The lower scores, the better matching. Therefore we approximately localize by the algorithm in Table II. In this way, the candidates are narrowed.

D. Accurate Localization

In order to evaluate the correspondence between two point clouds from different scans, Iterative Closest Point (ICP) algorithm[7] is available. ICP is an iterative method which tries to find the optimal transformation by minimizing the Euclidean distances error between the nearest points or surfaces. After the registration, the sum of those distances is minimized. It is equal to mean how two whole clouds are similar. It is available to evaluate the correspondence between not only the whole clouds but also between subsets in point clouds.

TABLE II
APPROXIMATE LOCALIZATION ALGORITHM

```
Algorithm Approximate_Localization$(M, m_{\mathrm{curr}})$ :
1:  $L := \emptyset$
2:  for every model $m_i \in M$ do
3:    $r :=$ Feature_Matching$(m_i, m_{\mathrm{curr}})$
4:    if $r \neq null$ do
5:      $(p, a, d) := r$
6:      if $a < a_{\mathrm{th}} \cap d < d_{\mathrm{th}}$ do
7:        $L := L \cup \{m_i\}$
8:      end if
9:    end if
10:  end for
11:  return $L$
```

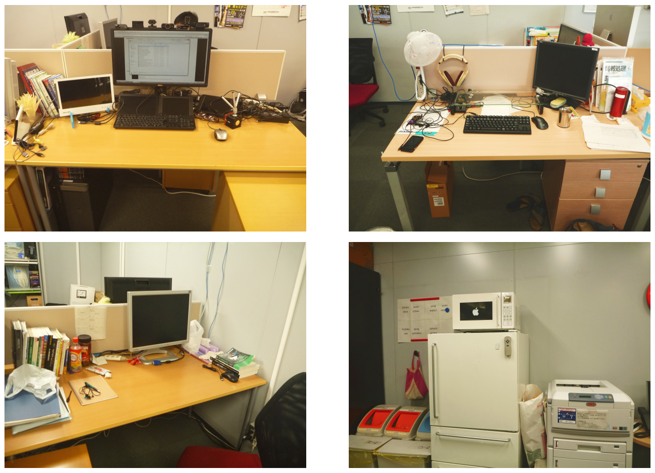

Fig. 9. Experimental Environments: Desk1 (left top), Desk2 (right top), Desk3 (left bottom) and Fridge(right bottom)

IV. EXPERIMENT

We performed an experiment to evaluate the basic idea of our method. Figure 9 depicts the experimental environment. The desks and the refrigerator were chosen as targets. Let us call "Desk1", "Desk2" and "Fridge". Three scenes for each environment were taken from left, front and right viewpoints, then processed and registered into the database. And then comparison with four new scenes from the front was performed. The new scenes are taken at three registered environments and one non-registered environment, say "Desk3". All objects were assumed to be fixed this time, and a few on planes. That is to say, desks are not so cluttered.

Table III presents the results of the approximate localization. The numbers of the planes n included in the models are depicted in round brackets. Void cells mean unmatched combinations since the criteria described in Line 4 in Table II was not satisfied. The marks X also mean unmatched because the criteria described in Line 6 in Table II was not satisfied. As shown in Table III, Desk1 and Desk2 are successfully localized. Furthermore, Desk3 is successfully rejected. The visualized results are depicted in Figure 10. However, some

TABLE III
RESULT OF LOCALIZATION: THE MODEL SCENES IN THE ROWS. THE TARGET SCENES IN THE COLUMNS. R_1 AND R_2 ARE MATCHED WHERE $R_1 = (5.86°, 0.102m)$ AND $R_2 = (2.88°, 0.034m)$.

		Desk1(5)	Desk2(6)	Desk3(5)	Fridge(3)
Desk1	Left(5)	X		X	
	Front(6)		X		
	Right(5)	R_1		X	
Desk2	Left(7)				
	Front(6)		R_2		
	Right(7)				
Fridge	Left(3)				X
	Front(4)				
	Right(3)				X
Estimate		Desk1	Desk2	None	None

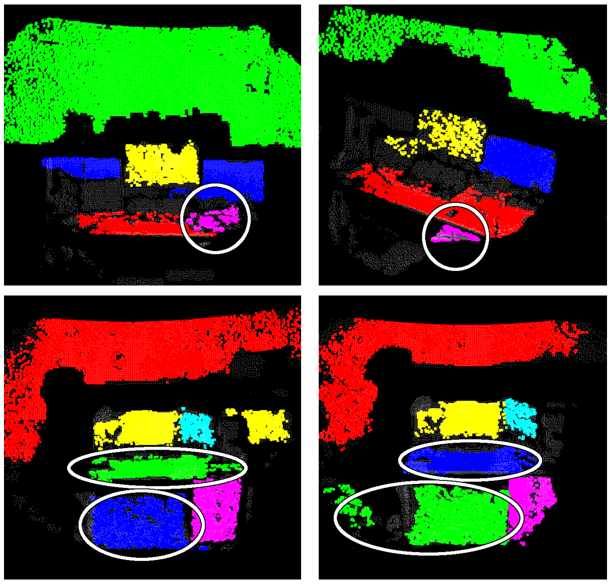
Fig. 10. Successful Results: Desk1 (top row) and Desk2 (bottom row), and The new scenes (left column) and the best matched models (right column). Planes in same colors correspond to each other. The purple planes are identified incorrectly in Desk1. The green and blue planes are detected upside down in Desk2.

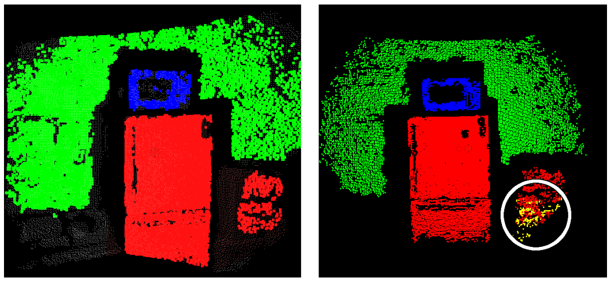
Fig. 11. Failed Results: The input scene (left) and the model which should be matched (right). The latter contains an incorrect plane colored yellow.

planes in Desk1 are not correctly identified. It is because that only two planes are parallel. Therefore they are detected upside down.

Fridge was not localized. It is because that a wrong plane was detected in the model as shown in Figure 11. We have to improve the plane extraction method.

V. CONCLUSION

In this paper, we proposed a localization method for markerless AR using a depth image sensor, and then evaluated our approximate localization part. We confirmed that our method could be used for a rough estimate. But there is room for improvement in this method. The plane extraction algorithm must works more precisely. We have to consider the usefulness even if the models in the database increase. For example, the current method requires an enough disk space because each scene has a point cloud of the same target from a little different viewpoints. To solve this problem, we would like to integrate them into one large point cloud. In this paper, we did not consider computing times. It is necessary to reduce the computational complexity for AR, for example by optimizing the code.

For accurate localization, ICP is available. We would like to verify it, and then implement an AR system in the future.

REFERENCES

[1] N. Kawaguchi, Small-sized Power Sensor and Wireless Display for Fine-grained Measurement and Presentation, *R'09 Twin and World Congress Poster session*, 2009.

[2] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, K. Tachibana, Virtual Object Manipulation on a Table-Top AR Environment, *In Proceedings of the International Symposium on Augmented Reality (ISAR)*, pp.111-119, 2000.

[3] G. Klein and D. Murray, Parallel Tracking and Mapping for Small AR Workspaces, *In Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pp.225-234, 2007.

[4] R. Castle, G. Klein and D. Murray, Video-rate Localization in Multiple Maps for Wearable Augmented Reality, *In Proceedings of the International Symposium on Wearable Computers (ISWC)*, pp.15-22, 2008.

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, S. Shotton, J. Hodges, D. Freeman, A. J. Davison, and A. Fitzgibbon, KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera, *In Symposium on User Interface Software and Technology (UIST)*, pp.559-568, 2011.

[6] R. B. Rusu, Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments, *PhD Thesis, Institut für Informatik der Technischen Universität München, Germany*, 2010.

[7] P. J. Besl and N. D. McKay, A Method for Registration of 3-D Shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 14, pp.239-256, 1992.