Recognition and Pose Estimation of Primitive Shapes from Depth Images for Spatial Augmented Reality

Ryo Hachiuma[1] and Hideo Saito[2]
University of Keio

ABSTRACT

In this paper, we propose a method for recognition and pose estimation of primitive shapes from depth images for spatial augmented reality (SAR). To use SAR in our everyday life, technology to recognize and estimate the pose of projected objects in the room is necessary. However, it is not a simple task to recognize primitive shapes because of their low 3D feature values. Hence, we focused on the gradient of normal vector map to extract surfaces and used the information of the surfaces of each object to recognize target objects. With our method, it becomes possible to recognize and estimate the pose of target objects in various scenes. Additionally, we projected an image onto each of the surfaces of the physical objects.

Keywords: spatial augmented reality, pose estimation, object recognition, RGB-D Camera.

Index Terms: mixed / augmented reality, object recognition

1 INTRODUCTION

Spatial augmented reality (SAR) is a technology to render arbitrary textures with the correct perspective onto the surface of 3D objects by projectors so that we can deal with virtual objects in a physical environment for 3D user interfaces. By replacing general purpose luminaires with image projectors, we can use SAR technologies for everyday life in our living room. One of the important technologies for adapting SAR to such everyday-life applications is recognition and estimation of poses of target objects in the living room so that the SAR system can render the required textures onto the target objects. Here, we focus on technology of detection, recognition, and localization of target objects by using cameras without putting any positioning sensors and markers on each target object. For achieving this purpose, we use a depth camera to capture 3D shape information of the target scene for recognition and pose estimation of the target object. Even from the depth images, target object recognition and pose estimation are not easy tasks, and they have been extensively studied for over 10 years. There are various approaches to the object recognition problem. For instance, Rusu et al. proposed point feature histograms (PFHs) as multi-dimensional features that describe the relationship of two normal vectors around a point for a 3D point cloud [7], while Tombari et al. proposed signatures of histograms of orientations (SHOT) as features that describe surface matching by computing the dot product of normal vectors [6]. According to such related investigations, we consider that using a normal vector map computed from the input depth image is very important for adapting arbitrary poses and positions of the target objects. Sano et al. [1] proposed a method of pose estimation of cubes for spatial augmented reality by efficient planar region detection using RGB-D superpixel segmentation.

In this paper, we propose a method for recognition and pose estimation from captured depth images by depth cameras such as Kinect. For efficiently performing the recognition and pose estimation procedure, we assume that the target objects consist of a set of primitive shapes, such as cubes, cuboids, cylinders, and pyramids. By such an assumption, we can define models of each object by a set of relative angles between neighboring sub-planar surfaces, which may be effectively segmented by images of normal vector directions of each point in the input depth images. According to such representation for each target object shape, the object shape may be recognized independently from its pose. Because the normal vector direction of each sub-planar surface can easily be estimated, the pose of the object can also easily be estimated. Since our method can recognize and estimate pose without using any texture information of the object surface, we can render arbitrary texture images onto white objects by projectors. By such projection, we can interactively change the rendered textures.

[1]ryo-hachiuma@hvrl.ics.keio.ac.jp
[2]hs@keio.jp

2 PROPOSED METHOD

2.1 Approach

As previously mentioned, to use the feature value of a normal vector to recognize an object, the object must be a complicated shape; therefore, objects that have a low number of feature points, like cubes, pyramids, and cylinders, are not able to be recognized in various scenes where one object gets on top of another. Our proposed method focuses on a surface that is a component of target objects. In this work, we use cubes, pyramids, cuboids, and cylinders as target objects. To recognize target objects, we store the surface information of each target object. For instance, in the case of a cylinder, we store the variance of the curved surface, the height, and the radius. In the case of a cube, the angle between surfaces and the length of the side is stored. We use this surface information to recognize each target object.

2.2 Overview

Our proposed method is divided into four main steps: 1) noise reduction, 2) surface segmentation, 3) object recognition using surface information, and 4) pose estimation. Figure 1 provides an overview of the process.

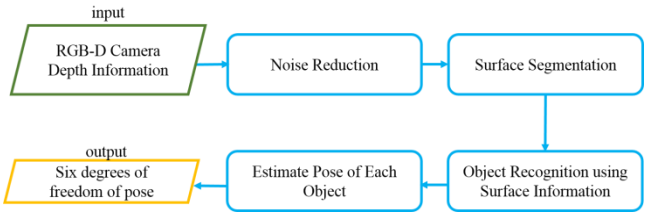


Figure 1: Overview of our proposed method

2.3 Noise Reduction

We use a Microsoft Kinect v.2 for the RGB-D camera in this investigation. The Kinect v.2 measures the depth value by ToF (time of flight). The ToF method acquires depth values by determining the time between the infrared light being emitted and the reflected light being detected. A ToF sensor uses two features of noise. The first is that the depth value near the edge of an object is inaccurate. The second is that the depth value at each frame is unstable. To reduce the first form of noise, a median filter is applied to the raw depth image information. Moreover, to reduce the second form of noise, we apply a temporal mean filter, averaging depth values of all individual pixels within n frames.

2.4 Surface Segmentation

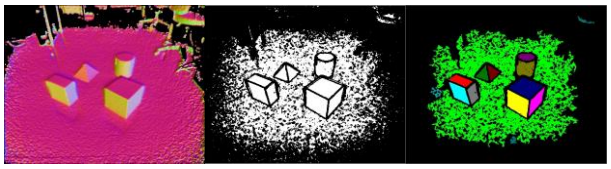
The second step of our method is to extract surfaces from raw depth images. The general approach to extract a plane from a 3D point cloud is to apply either a least-square method, PCA, or RANSAC algorithm to find the most likely parameters [4], [5]. However, when trying to extract a surface using all points, the points near object edges are likely to be recognized as the wrong surface. Therefore, our method uses the idea from Uckermann et al.'s method [2]. We take three steps to extract a surface from a raw depth image.

In the first step, we compute the normal vector of each 3D point. Four neighborhood points in the depth image are considered to compute a normal vector. We compute four classical cross products of a point: left and up, up and right, right and down, and down and left. We then average these four computed vectors and repeat this process for all points to obtain normal vectors for all points. Figure 2(a) is an image depicting the direction of the normal vectors XYZ that are converted into a RGB pseudo-color space.

The second step focuses on the gradient of normal vector map; we compute the scalar product of normal vectors. We look for eight neighborhood pixels of a point. At each point and its neighborhood pixel, we compute the scalar product of normal vectors. The pixel value of each point is calculated by averaging eight scalar products. After calculating each pixel value, the image is converted into a binary image by using the threshold value $\theta_{max} = 5.5°$. Figure 2(b) illustrates the gradient image of the normal vectors.

For the last step of segmenting surfaces from the gradient image, we apply a region growing algorithm. If the conditions of eight neighborhood pixels of a point are correct, the region growing algorithm labels the pixels as the same region as the point. This process is repeated until all of the pixels are labeled. Figure 2(c) shows a set of segmented surfaces with each surface point represented by a unique patch ID.

To facilitate the object recognition phase, we compute the equation of each surface. We use the RANSAC algorithm to calculate the parameter. The RANSAC algorithm is a method to calculate all the parameters in the data except outliers.



(a) Normal vector map (b) gradient image (c) segmented image

Figure 2: Surface extraction using gradient image: (a) direction of normal vectors XYZ are converted into RGB pseudo-color space, (b) using eight neighborhood pixels, dot product is computed and binarized by threshold value, and (c) surface segment image is created using region growing algorithm.

2.5 Object Recognition

In this section, the surface segmented image from the previous phase is used, and we recognize each target object from the component surfaces that make up a target object. As previously mentioned, we recognize the target object with already known surface information. We take three steps to find each object from the segmented image. The first step is focusing on the variance of surfaces to recognize cylinders, the second step is focusing on the angles between neighborhood surfaces to recognize pyramids, and the last step is focusing on the Euclidean distance to recognize cubes and cuboids.

In the first step, the variance of normal vectors whose points are composed of the curved surface of a cylinder is larger than that of any other surfaces. Hence, we compare the variance of the side of the cylinder from surface information with that of the surface extracted in Section 2.4.

In the second step, we recognize pyramids and other shapes (cubes, cuboids). In this step, we focus on the angles between surfaces that exist in the neighborhood of the shapes. In this work, the lengths of the sides of the target objects are within 20 cm. After calculating the centroid of each surface using the equation of surfaces computed in the previous step, we compute the angle between surfaces only when the distance between centroids are within 20 cm. Finally, we compare the angles between the surfaces with the angles of surface information we stored before. To do this, we determine which surfaces are components of pyramids and cubes from the surface segment image. In this step, we do not recognize cubes and cuboids.

In the last step, we recognize cubes and cuboids from the surface segment image. If we capture the scene with one RGB-D camera, it is difficult to capture four surfaces of a cube. Hence, this step will run only when the number of surfaces recognized as a cube in the previous step is more than four. Three steps are taken to recognize cubes and cuboids using Euclidean distance. In the first step, we divide the group of surfaces into two groups arbitrarily. In the next step, we calculate the centroid of each group of surfaces, which is defined in the previous step. In the last step, we calculate the distance d_i between the centroids of two groups.

2.6 Pose Estimation

After recognizing each object, we finally estimate each object's 6DoF pose. To estimate it, we have to compute the transform matrix from an object coordinate that is defined at each target object transform it to a camera coordinate. Figure 3 depicts the pose estimation. The transform matrix consists of a rotation matrix and translation vector.

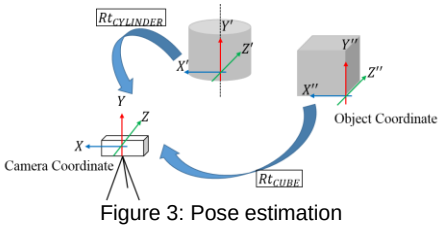


Figure 3: Pose estimation

The transform matrix of the polyhedron is uniquely determined. In contrast, the transform matrix of the cylinder is not uniquely determined. Therefore, the transform matrix must be calculated at each shape.

We compute the rotation matrix of the polyhedron. In this work, we take particular note of normal vectors of surfaces. To compute the rotation matrix, the correspondence relationship of three normal vectors is needed. Therefore, two normal vectors of the polyhedron's surfaces and a normal vector of the floor are chosen to compute the rotation matrix. Next, we compute the translation vector of the polyhedron. To compute it, we find the origin of each coordinate at the camera coordinate. The origin is defined as the point of intersection among two side surfaces of each polyhedron and the floor. However, as the case where the target object gets on top of the other target object is included in this work, the translation vector using the floor surface cannot be computed correctly. Hence, the centroid of each object is calculated by the centroids of the surfaces to determine whether the target object gets on top of another object. If it gets on top of another, we re-calculate its translation vector by translating the surface floor toward the direction of the floor's normal vector.

After that, we compute the rotation matrix of the cylinder. The three rotational components are needed to calculate the rotation matrix. The first component is the direction of the cylindrical axis. The second component is the vector that is parallel to the tangent line of the floor surface. The last component is a vector that is the cross product of the first and the second component vector. Using these three vectors, the rotation matrix is calculated in the same way as for the polyhedron. Next, we compute the translation vector of the cylinder. Like the polyhedron's translation vector, the origin of the object coordinate is found to compute the translation vector of the cylinder. The origin is defined at the intersection point of the cylindrical axis and its base. However, that point cannot be captured by camera. Hence, we compute that point using the side curved surface. As we look at a point of the side, we translate that point toward the opposite direction of its normal vector. Then, all of the points are in the direction of the cylindrical axis. Finally, the centroid is computed using these points and is translated to the base.

3 EXPERIMENT

Three experiments were conducted to show the effectiveness of our method. First, we displayed the object coordinates that were defined by each object to show the result of pose estimation. Second, we projected the shape line of each object using a projector to show the result of object recognition. Finally, we evaluated the pose estimation of our method. All of the experiments were performed with the following system: CPU – Intel Core i7-4790 3.60 GHz, RAM – 8.00 GB, RGB-D camera – Microsoft Kinect v2.0 (512 x 424), and projector – Epson EB-W8 (1280 x 1024). Figure 4 depicts our experimental setup.

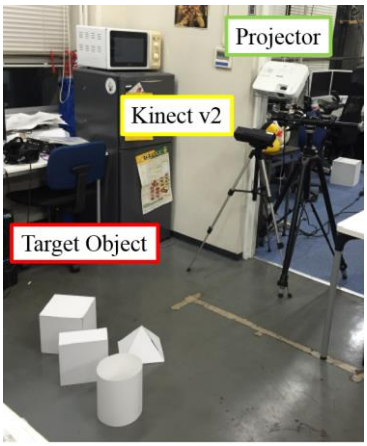


Figure 4: Experimental setup

3.1 Result of Pose Estimation

We experimented on qualitative evaluation of pose estimation in the scene where four target objects are set on the floor or one objects gets on top of another object in various scenes. To evaluate quality, we displayed each object coordinate. Figure 5 depicts the result of pose estimation.

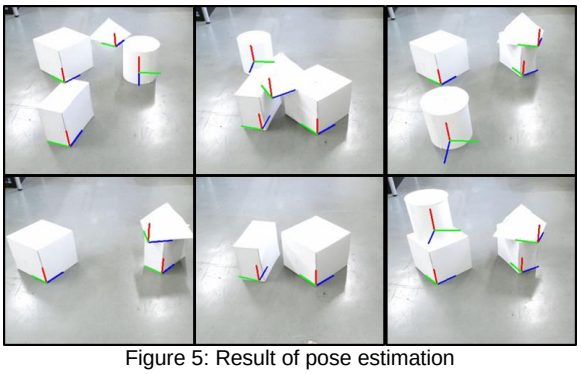


Figure 5: Result of pose estimation

It is shown in Figure 5 that the pose of each target object was estimated in the case of objects that are set on the floor or one object gets on top of another in various ways. For instance, pose was estimated in the case where the cylinder got on top of the cube and the case where the pyramid got on top of the cuboid, as shown in Figure 5, bottom right.

3.2 Result of Recognition

In this section, we projected the image at each target object to see the result of recognition and pose estimation. Figure 6 illustrates the result of object recognition and pose estimation using our proposed method.

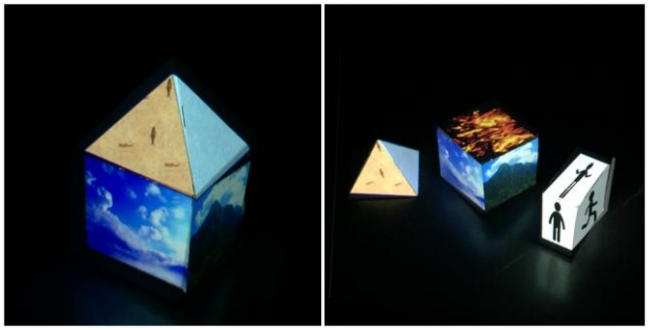


Figure 6: Result of object recognition

3.3 Accuracy Evaluation

To evaluate the pose estimation of our proposed method, we compared the rotation matrix and translation vector with ground truth.

We used the Riemannian distance to calculate the rotation error between two rotation matrices. The Riemannian distance between two rotation matrices has been defined by Moakher [3]. Moreover, we used the Euclidean distance to calculate the translation error between two translation matrices. Figure 7 illustrates the average error of pose estimation at each object.

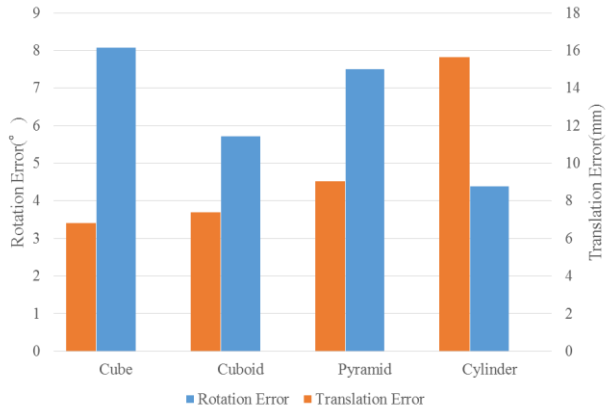


Figure 7: Average Error of Pose Estimation

As shown in Figure 7, the translation error of each object fell within the range of 1 cm except for the cylinder, and the rotation error of each object fell within the range of 10° at all target objects. The reason for the translation error is the rotation matrix of the polyhedron was calculated by its surfaces, but the rotation matrix of the cylinder was calculated by its curved surface points. This may cause an error of approximately 1.4 cm in the translation vector.

4 Limitation

In this section, we state the limitation of our proposed method. As we previously mentioned, we used two side surfaces to compute the shape's translation vector. Hence, in the case where two side surfaces are not able to be detected (e.g. cube and cuboid connected horizontally, or target object facing directly forward to the camera), the translation vector is not calculated by our proposed method. To overcome this limitation, we have to calculate the translation vector using two voluntary surfaces of target objects.

5 Future Plan

We are planning on implementing a number of extensions to improve the system. First, the current system is restricted to a single copy of each of the four shapes. In the future, the system would be able to process multiple copies of each of the physical shapes. This capability would greatly enhance the functionality of the system. We would also like to extend the range of shapes the system can recognize. Second, we would like to configure the system to operate as a standalone tracking system. This will enable the system to operate with other rendering systems and applications. We will investigate the use of a common tracker abstraction mechanism, such as VRPN.

Finally, we would like to evaluate the tracking technique in relation to a real world application. We are targeting the domain of industrial design prototyping. The unadorned white shapes lend themselves to rendering and allow designers to configure multiple potential design concepts. Textures for the shapes could be either rendered from the designers' existing suite of design applications or by a custom application that could be developed to allow the designers to directly adorn the white shapes with virtual paint. The physical nature of the white shapes allows for the designers to quickly configure multiple design concepts. Working with industrial designers inspired this future application vision.

6 Conclusion

We proposed a method to estimate the pose of target objects for spatial augmented reality. In our proposal, depth data from an RGB-D camera is divided into surface regions by focusing on the gradient of normal vector map. Using surface information, we recognize each target object. Finally, we estimate the pose of each target object in various scenes where one shape gets on top of another.

Acknowledgements

This work was supported in part by JSPS Grant-in-Aid for Scientific Research(S) 24220004.

References

[1] Masayuki Sano, Kazuki Matsumoto, Bruce H. Thomas, Hideo Saito, Rubix: Dynamic Spatial Augmented Reality by Extraction of Plane Regions with a RGB-D Camera, Proceedings of 2015 IEEE International Symposium on Mixed and Augmented Reality, pp. 148–151, 2015.

[2] Andre Uckermann, Robert Haschke, and Helge Ritter. ``Real-time 3D Segmentation of Cluttered Scenes for Robot Grasping." Humanoid Robots (Humanoids 2012), pp. 198–203, 2012.

[3] Maher Moakher. ``Means and Averaging in the Group of Rotations." SIAM journal on matrix analysis and applications 24.1 pp. 1–16, 2002.

[4] Edward Castillo, Jian Liang, and Hongkai Zhao. ``Point Cloud Segmentation and Denoising via Cnstrained Nonlinear Least Squares Normal Estimates." Innovations for Shape Analysis. Springer Berlin Heidelberg, pp. 283–299, 2013.

[5] Radu B. Rusu, et al. ``Close-Range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Domestic Environments." Intelligent Robots and Systems, (IROS 2009), pp. 1 6, 2009.

[6] Federico Tombari, Samuele Salti, and Luigi Di Stefano. "Unique signatures of histograms for local surface description." *Computer Vision–ECCV 2010*. Springer Berlin Heidelberg, pp. 356–369, 2010.

[7] Radu B. Rusu, et al. "Aligning point cloud views using persistent feature histograms." Intelligent Robots and Systems, (IROS 2008), pp. 3384–3391, 2008.