Object Placement Planner for Robotic Pick and Place Tasks

K. Harada, T. Tsuji, K. Nagata, N. Yamanobe,
H. Onda, T. Yoshimi, and Y. Kawai

***Abstract*— This paper proposes an object placement planner for a grasped object during pick-and-place tasks. The proposed planner automatically determines the pose of an object stably placed near a user assigned point on an environment surface. The proposed method first constructs a polygon model of the surrounding environment, and then clusters the polygon model of both the environment and the object where each cluster is approximated by a planar region. The placement of the object can be determined by selecting a pair of clusters between the object and the environment. We further impose several conditions to determine the pose of the object placed on the environment. We show that we can determine the position/orientation of the object placed on the environment for several cases such as hanging a mug cup on a bar. The effectiveness of the proposed research is confirmed through several numerical examples.**

I. INTRODUCTION

The pick-and-place is one of the most typical tasks for a robot required to achieve. However, it is often difficult for a robot to automatically plan the pick-and-place motion. One reason for this difficulty is the complexity of the geometry of both the environment and the grasped object. Fig. 1 shows a typical case of a robot's working environment for achieving a pick-and-place task. When a robot performs the pick-and-place task in an environment where many daily objects are randomly placed, the robot may place the object in a narrow area surrounded by other objects or may on top of other objects. Sometimes the robot may hang an object on a bar. However, it is not clear how to determine the position/orientation of an object stably placed on the environment.

To deal with this problem, this research proposes an autonomous object placement planner that determines the stable position/orientation of an object placed on the environment. Our planner first triangulates the captured point cloud data of the environment. Then, it clusters the polygon models of the object and the environment. Each cluster can be approximated by a planar region. The object placement planning is performed by selecting a pair of clusters where one is from the object and the other is from the environment. When selecting a pair of clusters, we consider the shape's convexity since the concave part of the object surface cannot be in contact with the concave part of the environment surface. Then, we perform the *inclusion test* and the *stability test* checking whether or not the object can be stably put on the environment surface.

K.Harada, K.Nagata, N.Yamanobe, H. Onda, T. Yoshimi and Y.Kawai are with Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan kensuke.harada@aist.go.jp

T.Tsuji is with the Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan tsuji@ait.kyushu-u.ac.jp

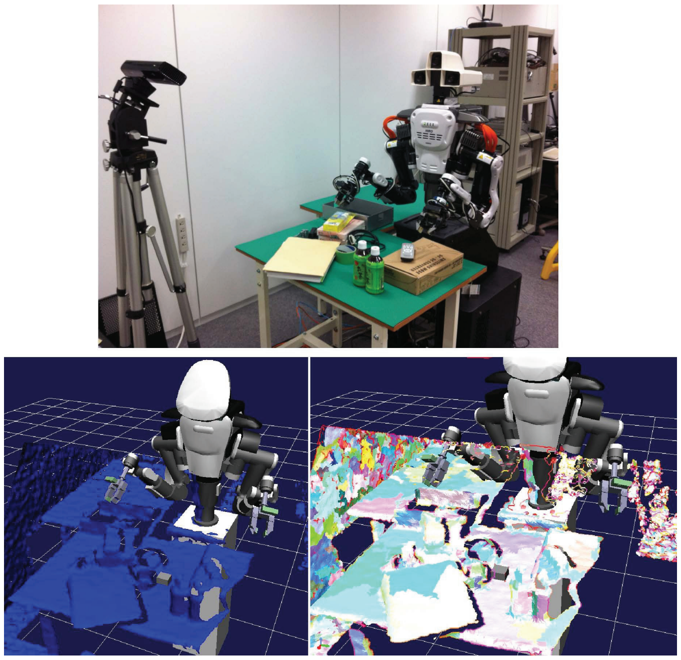
Fig. 1. Working environment to perform the pick-and-place task

The authors have already proposed a grasp planner for parallel grippers[1] where the posture of the hand stably grasping the object is computed by considering the finger surface elasticity. This method is effective since we can estimate the contact area between the finger and the object and can plan a grasping posture with a large contact area. Then, by using this grasp planner, the authors have proposed a pick-and-place planner for dual arm manipulators[2]. In this research, the object placement planner was briefly outlined as a part of the whole pick-and-place planner. Also, this placement planner could just be applied for a limited case of the environment shape. On the other hand, this research extends and generalizes the object placement planner and applies it to several examples.

This paper is organized as follows; after introducing the related works in Section 2, Section 4 details the offline surface clustering method. Then, Section 5 details the searching method of the object pose. Finally, Section 6 demonstrates the efficiency of our method through several numerical examples.

II. Related Works

Lozano-Perez et al.[3] proposed the grasp and motion planning problem. Then, during a decade, the grasp planning problem has been extensively researched[4], [5], [6], [7], [8], [9], [1]. As for the object placement planner in cluttered environment[10], [11], Berenson et al.[6] planned the grasping posture of a multi-fingered hand taking the putting posture into consideration. Here, the putting posture of the object is assumed to be fixed. Recently, Cosgun et al.[12] proposed the push planning on a table with many obstacles. Schuster et al.[13] used the point cloud and identified a a planar area on a table. On the other hand, our resaerch deal with more general cases where the geometrical relation between an object and an environment is considered.

There are several research on identifying a planar area from the poinc cloud data such as[14], [15], [18]. Clustering method of polygon models has been extensively researched in the research community of computer graphics such as[16], [17]. On the other hand, this research extends the clustering method for object placement planning by robotic manipulators.

III. Nomenclature

We assume that the polygon model of the grasped object is given. As for the surrounding environment of the robot, we assume that the point cloud data is obtained and is transformed into the polygon model. For each polygon model, we define the following variables:

C_i $(i = 1, \cdots, n)$: i-th cluster of the polygon model as a set of triangles.
P_i $(i = 1, \cdots, n)$: Plane fitted to C_i.
n_i $(i = 1, \cdots, n)$: 3D unit outer normal vector of P_j.
p_i $(i = 1, \cdots, n)$: 3D position vector of a point on P_j.
p_{gi} $(i = 1, \cdots, n)$: 3D position vector of the CoG (center of gravity) of the point set included in C_i.

IV. Offline Surface Clustering

This section describes the offline clustering method of the polygon models of the object and the environment. We will briefly explain the overview of the basic algorithm of the clustering method since its detail has already described in the previous papers[1], [2]. In the clustering algorithm, we first calculate the initial set of clusters where each cluster is composed of a few triangles. Then, we consider iteratively merging a neighboring cluster as far as the cluster can be approximated by a planar region. Fig. 2 shows the clustered models of the grasped objects: pet-bottle, dog, mug-cup and duck. Also, the clustered model of the environment is shown in Fig. 1.

When searching the object posture, we consider that the surface of the object maintains contact with the surface of the environment. For this purpose, we consider selecting a pair of clusters where one is from the object and the other is from the environment. For the purpose of searching the object posture, we calculate the following parameters for each cluster:

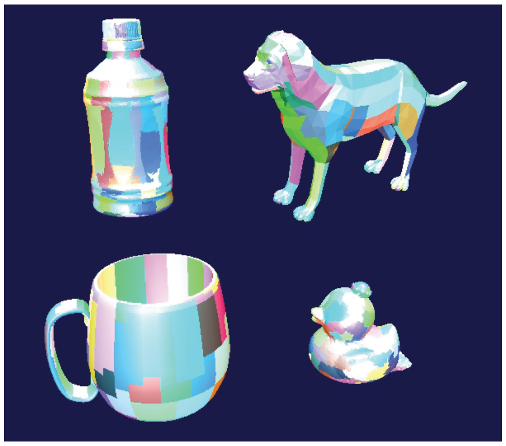

Fig. 2. Clusters of the object models

Definition 1 Cluster Parameters

Fitted Plane: Normal vector n_i of P_i and the position vector p_{gi} projected onto P_i are calculated.

Boundary: Vertices included in more than two clusters are defined as the boundary vertices. By using the lines including the boundary vertices, we define the boundary of clusters(Fig. 3). We define the outer and the inner boundaries.

Contacting Cluster: The boolean parameter **Contacting Cluster** is TRUE for the candidate cluster for making contact.

Convexity: The parameter expressing the convexity; CONVEX(CV), CONCAVE(CC) and NOTCLASSIFIED(NC) is assigned for each cluster.

As for the **Fitted Plane**, we set the fitted plane of the object's cluster coinciding with the fitted plane of the environment's cluster when searching the object pose. As for **Contacting Cluster** and **Convexity**, the details are explained in the following subsections.

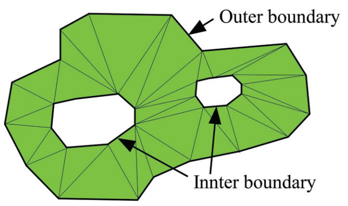

Fig. 3. Outer/inner boundaries of a cluster

A. Contacting Clusters

Since a large number of clusters are calculated especially for complex shaped polygon models, it is not reasonable to search all the clusters in real-time. Also, the user may want to assign the point on the environment where the object is placed. Hence, the boolean parameter **Contacting Cluster** is associated for each cluster. If a cluster is a candidate for making contact, the parameter **Contacting Cluster** is set to be TRUE.

We now explain how to set this parameter in this paper. For the polygon model the object, we manually click multiple points on the object surface appeared in the graphics window. The parameter **Contacting Cluster** of clusters including the clicked point is set to be TRUE. For the polygon model of the environment, we manually click a single point just before searching for the object pose. The parameter **Contacting Cluster** of a cluster including the clicked point is also set to be TRUE. The robot performs the pick and place task so that the grasped object is stably placed near the clicked point on the environment.

B. Convexity

We calculate the convexity of each cluster. For the cluster C_i, let C_{ij} $(j = 1, \cdots, m_i)$ be the neighboring clusters. We set the position vector p_i and p_{ij} so that the lines L_i: $p_i + t_i n_i$ and L_{ij}: $p_{ij} + t_{ij} n_{ij}$ have an intersection. At the intersection point, we check the sign of the scalar values t_i and t_{ij}. Now we have the following definition:

Definition 2 Convexity

CONVEX (CV) : If $t_i < 0$ and $t_{ij} < 0$ are satisfied for all the neighboring clusters, the cluster C_i is defined to be CONVEX.

CONCAVE (CC) : If $t_i > 0$ and $t_{ij} > 0$ are satisfied for all the neighboring clusters, the cluster C_i is defined to be CONCAVE.

NOTCLASSIFIED (NC) : If the sign of t_i and t_{ij} change depending on the neighboring clusters, the cluster C_i is define to be NOTCLASSIFIED.

C. Table-Leg Clusters

When we calculate the convexity of clusters, special attention should be paid for the objects having multiple legs such as table, chair, human figure, and animal figure (Fig. 4). These objects usually stand on the ground by the sole of the foot. However, by simply clustering the polygon model, each sole of the foot belongs to the different clusters. Now we have the following definition:

Definition 3 Table-leg cluster

Table-leg cluster is a cluster composed of multiple separated areas

To obtain the table-leg cluster, we calculate the following equations for all the combination of two clusters of the polygon model. For the clusters C_i and C_j we calculate

$$\begin{aligned} l_{ij} &= (p_i - p_j) \cdot n_i, \\ m_{ij} &= 1 - n_i \cdot n_j, \ (i, j = 1, \cdots, n). \end{aligned}$$

If the following three conditions are satisfied at the same time, we consider merging C_j to C_i:

1. l_{ij} and m_{ij} are smaller than the pre-defined threshold,
2. The parameter **Convexity** of the clusters C_i and C_j are both CONVEX, and
3. C_j is not the neighboring cluster of C_i.

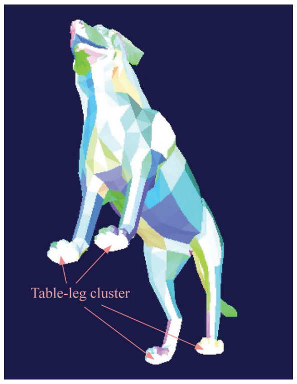

Fig. 4. Clusters at the sole of the foot merged into a single cluster

V. OBJECT PLACEMENT SEARCH

This section explains the searching method of the object pose. With our current setting, we search the object pose online as a part of the pick-and-place motion planning. However, we can search the object pose offline as long as there is no significant change in geometry of the environment. Our object placement planner first obtains some pairs of clusters. For the pairs of clusters, we check the convexity of the clusters. Then, we check whether or not the cluster is included in the outer boundary of its pair cluster projected on the common fitted plane. We further check the static stability of the object.

A. Convexity Test

When we consider that the surface of the object contacts with the surface of the environment, we can classify the contact states based on the combination of the cluster's convexity parameter as shown in Fig.5. We can assume 9 states; CV-CC, CV-CV, CC-CV, CC-CC, NC-CV, CV-NC, NC-CC, CC-NC and NC-NC. By using this classification, we can obtain a necessary condition for a pair of clusters to maintain contact. For example, the contact cannot be maintained if the state is CC-CC. This is because a concave part of the object surface cannot maintain contact with a concave part of the environment surface. Based on this consideration, we can classify the contact states into the **Applicable** and **Not-applicable** ones. We have the following theorem:

Theorem 1 (Convexity Test)

For a pair of clusters to contact each other, the convexity parameters have to be classified as **Applicable** *shown in the following cases*:

Applicable: CV-CC, CV-CV, CC-CV, NC-CV, CV-NC, and NC-NC

Not Applicable: CC-CC, NC-CC, and CC-NC

By using this classification, we can kick out three contact states and can speed up the object placement planning.

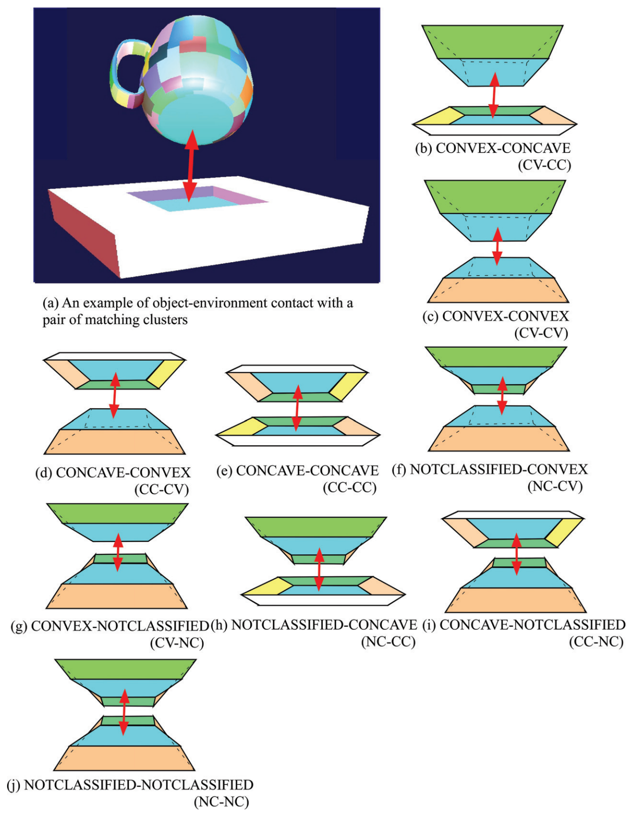
(a) An example of object-environment contact with a pair of matching clusters
(b) CONVEX-CONCAVE (CV-CC)
(c) CONVEX-CONVEX (CV-CV)
(d) CONCAVE-CONVEX (CC-CV)
(e) CONCAVE-CONCAVE (CC-CC)
(f) NOTCLASSIFIED-CONVEX (NC-CV)
(g) CONVEX-NOTCLASSIFIED (CV-NC)
(h) NOTCLASSIFIED-CONCAVE (NC-CC)
(i) CONCAVE-NOTCLASSIFIED (CC-NC)
(j) NOTCLASSIFIED-NOTCLASSIFIED (NC-NC)

Fig. 5. Classification of contact states between the object and the environment based on the convexity parameter.

B. Cluster Inclusion Test

By using the pairs of clusters found out to be **Applicable** in **Theorem 1**, we will obtain candidates of the object pose contacting with the environment surface. For this purpose, we perform the *cluster inclusion test* explained in this subsection. The overview of the cluster inclusion test is shown in Fig.6. Let us consider the case where a pair of clusters shares a common fitted plane. Also, this subsection considers the case where all the vertices of the cluster are projected onto their fitted plane.

Although the pose of the object is arbitrary as far as the object's cluster shares a common fitted plane with the environment's cluster, this paper assumes eight poses of the object. We first consider two cases of the relation between the CoG of the object's cluster (abbreviated as OCoG) and the CoG of the environment's cluster (abbreviated as ECoG). In the first case, position of the OCoG coincides with the position of the ECoG. And in the second case, the position of the OCoG coincides with the user assigned point on the environment surface. For these cases, we consider obtaining candidates of the object pose by rotating the object about the normal of the fitted plane for 0, 90, 180, and 270 [deg]. Fig.6 shows four of eight object poses.

Since Fig.6 shows an example of the CV-CC contact, all the vertices of the object's cluster should be included inside the outer boundary of the environment's cluster. If this condition is not satisfied, the object cannot maintain contact with the environment. On the other hand, if this condition is satisfied, we express that CV $\subset$ CC is satisfied. Now we have the following theorem:

Theorem 2 (Cluster Inclusion Test)

For the contact states found out to be **Applicable** except for CV-CV case, one of the conditions; CV $\subset$ CC, CC $\supset$ CV, NC $\supset$ CV, *and* CV $\subset$ NC *should be satisfied.*

Among four object poses shown in Fig.6, CV $\subset$ CC is satisfied only for the case B.

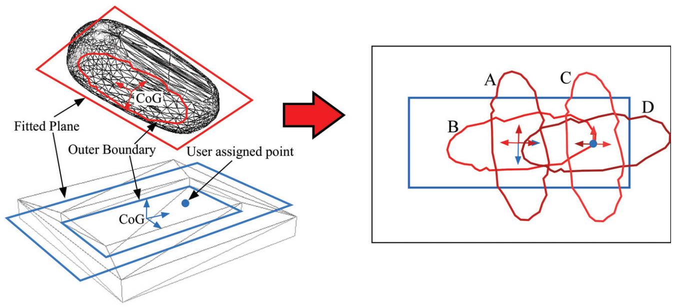

Fig. 6. Inclusion test for a pair of clusters where CV $\subset$ CC is satisfied for the case B

C. Stability Test

For the pose of the object satisfying **Theorem 2**, we check whether or not the pose is statically stable. As in the previous subsection, let us consider the case where a pair of clusters shares a common fitted plane. Also, we consider the case where all the vertices of the cluster are projected onto the fitted plane.

Assuming that the friction between the object and the environment is large enough, we check whether or not the vertical line including the object's CoG passes through the supporting area. The overview of the algorithm is shown in Fig. 7. This algorithm is summarized as follows:

Algorithm 1 (Stability Test)

Step 1: Set a pair of clusters sharing a common fitted plane and project the inner/outer boundaries onto the fitted plane.

Step 2: Obtain the common area included in the outer boundaries.

Step 3: From the common area obtained in Step 2, exclude the area included in the inner boundaries.

Step 4: For the area obtained in Step 3, calculate the 2D convex hull.

Step 5: If the vertical line including the object's CoG passes through the 2D convex hull, the contact is stable.

If the condition of **Step 4** is satisfied, we judge that the pose of the object is statically stable under the gravitational field. Here, before executing this algorithm, we check the intersection between the vertical line including the CoG and the fitted plane. If vertical position of the intersection is larger than the CoG, we judge that the contact is stable and do not execute Algorithm 1. This is because, even if the contact is not stable by using Algorithm 1, we can expect that the contact will converge to a stable state.

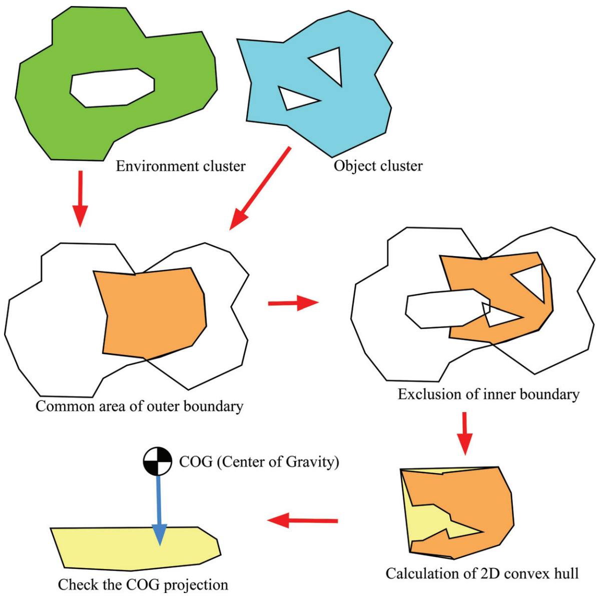
Environment cluster
Object cluster
Common area of outer boundary
Exclusion of inner boundary
COG (Center of Gravity)
Check the COG projection
Calculation of 2D convex hull

Fig. 7. Stability checking algorithm

D. Integration to Pick-And-Place Planner

By using the proposed method, we prepare multiple candidates of object pose by applying the **Theorem 2** and check the static stability for each object pose by using **Algorithm 1**. Hence, the proposed object placement planner may generate multiple candidates of the object pose. In our setting, we set the higher priority to the object pose where the CoG of the object's cluster coincides with the user assigned point on the environment surface projected onto the fitted plane.

We execute the pick-and-place planner[2] for each object pose. If we obtain the feasible pick-and-place motion, we will terminate the planner. Here, we note that Theorem 1 and 2 just give necessary conditions for two clusters sharing a common fitted plane. When performing the pick-and-place planner, we check whether or not the object can really be placed on the assigned part on the object surface by checking the collision between the object and the environment.

VI. RESULTS

We confirm the effectiveness of the proposed approach by numerical examples. We used the model of the dual-arm manipulator HiroNX having 2DOF head, two 6DOF arms, and 1DOF waist. As shown in Fig.1, we obtained the point cloud data of the environment by using the Kinect sensor and transformed it to the polygon model by using the Point Cloud Library[18]. We coded the pick-and-place planner as a plugin of Choreonoid [20].

We first measured the calculation time needed to triangulate and clusterize. We used the PC with 3.2GHz CPU.

TABLE I
CALCULATION TIME NEEDED FOR TRIANGULATION AND CLUSTERIZATION

Model	Triangles	Triangulation [s]	Clusterization [s]
Mug cup	3450		0.2
Duck figure	13550		1.14
Dog figure	3220		0.18
Environment	65371	6.0	18.0

Table I shows the result of calculation where the name of the model, the size of triangles, the time needed for triangulation and clusterization. Here, the VRML model of the dog figure was obtained from the Princeton Shape Benchmark[19]. With the current setting, we wait for 24[s] to start the object placement planner.

Then we show the calculation result of the pick-and-place motion. Fig.8 shows the pick-and-place motion of the dog figure. The red arrow shows the assigned point on the environment. For the model of the object, the parameter **Contacting Cluster** of the cluster at the foot sole is set to be TRUE. This cluster is a typical example of the table-leg cluster as is explained in this paper. It took about 0.5[s] to calculate the pick-and-place motion. As shown in the figure, the robot successfully performs the pick-and-place motion and place the object at the assigned point on the environment surface.

Fig.9 shows the pick-and-place motion of a mug cup. Through this example, we consider hanging a handle of the cup to a bar. In this example, since the vertical position of the intersection between the gravity vector and the fitter plane is larger than the object's CoG, we did not check Algorithm 1. Also, the parameter **Contacting Cluster** of the cluster inside the handle part is set to be TRUE. As shown in the figure, the robot successfully hangs the handle of the cup to the bar.

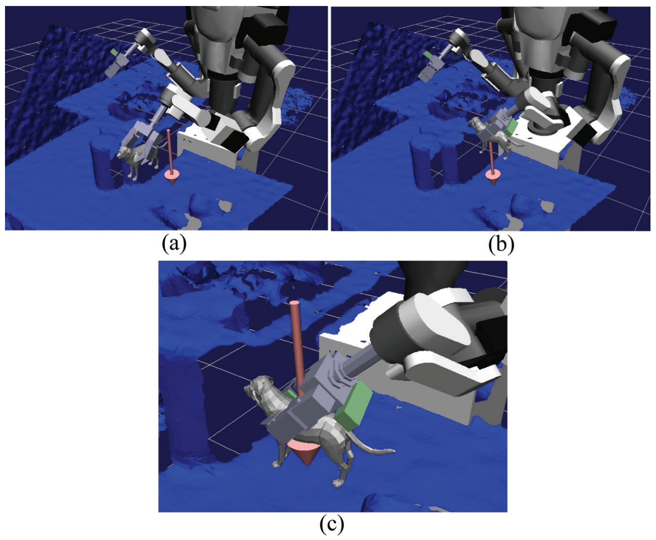

Fig. 8. Pick-and-place motion of dog figure

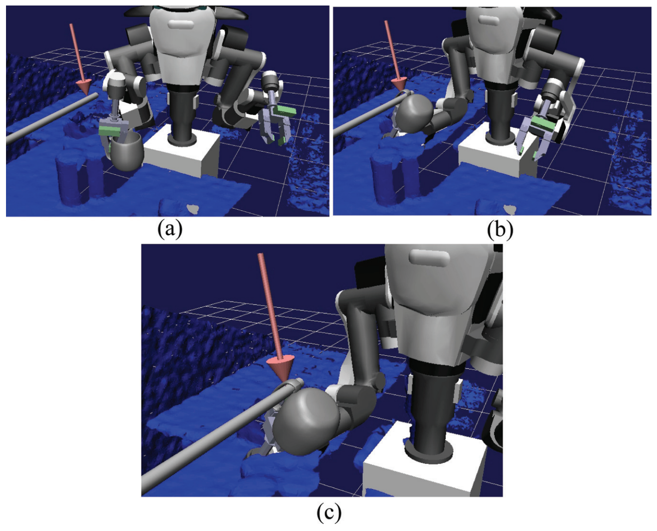

Fig. 9. Pick-and-place motion of mug cup

VII. DISCUSSION

A. Polygon Model of Environment

In this research, we construct the polygon model of the environment and clustered it. By using the polygon model of the environment, we can easily check the collision by using standard collision checkers. On the other hand, the triangulation of point cloud takes about 6[s]. Here, since Point Cloud Library[18] prepares a basic method of plane segmentation of point cloud, we can reduce the calculation time by directly using the point cloud data in the object placement planner. Also, we may not need to clusterize whole surface of the environment model. Application of our method to dynamically changing environment is a future research topic. Furthermore, the difference between the real environment and its polygon model should be taken into consideration.

B. Inner and Outer Boundaries

For the polygon model of an object, it is easy to define the outer and the inner boundaries of a cluster since the order of vertices included in a triangle is uniform and there is no break of polygons. However, for the case of the polygon model of an environment, it sometimes becomes difficult to define the outer and the inner boundaries of clusters. As for the polygon model of the environment used in this research, instead of using the outer boundary, we calculated the 2D convex hull of points included in the cluster by using qhull[21]. Also, we did not consider the inner boundaries of the clusters of the environment model.

VIII. CONCLUSIONS

This paper proposed a method for object placement planning during the pick-and-place motion of robot manipulators. In our method, we first cluster the polygon model of the object and the environment. Then, candidates of object pose placing near the assigned point on the environment surface are obtained through several tests such as the *convexity test*, the *cluster inclusion test* and the *stability test*. Through numerical examples, we confirmed that the object can be stably placed on the surface of the environment model captured by using the Kinect sensor.

The authors would like express their sincere gratitude for Mr. Michiharu Asahi with SFC Co. Ltd. for his help in setting up the Kinect sensor and Dr. Juan Rojas for helpful discussions. Lastly, this research was conducted under the support of the NEDO intelligent robot technology software project. Lastly, the source code of the pick-and-place planner used in this research is available from [20].

REFERENCES

[1] K. Harada et al., “Grasp Planning for Parallel Grippers with Flexibility on its Grasping Surface”, *Proc. of IEEE Int. Conf. on Robotics and Biomimetics*, 2011.
[2] K. Harada et al., “Pick and Place Planning for Dual-Arm Manipulators”, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2012.
[3] T. Lozano-Perez, et al., “ HANDY: A Robot System that Recognizes, Plans, and Manipulates, ”*Proc. of IEEE lnt. Conf. on Robotics and Automation*, 1987.
[4] C. Borst, M. Fischer, G. Hirzinger, “A fast and robust grasp planner for arbitrary 3D objects”, *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp.1890-1896, 1999.
[5] A.T. Miller et al., “Automatic Grasp Planning using Shape Primitives”, *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 14-19, 2003.
[6] D. Berenson, J. Kuffner, and H. Choset, “An Optimization Approach to Planning Mobile Manipulation”, *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp.1187-1192, 2008.
[7] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, “BiSpace Planning: Concurrent Multi-Space Exploration”, *Proc. of Int. Symposium on Robotics, Science and Systems*, 2008.
[8] K. Harada, K. Kaneko, and F. Kanehiro, “Fast Grasp Planning for Hand/Arm Systems Based on Convex Model”, *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1162-1168, 2008.
[9] T. Tsuji, K. Harada, K. Kaneko, F. Kanehiro, and K. Maruyama, “Grasp Planning for a Multifingered Hand with a Humanoid Robot”, J. of Robotics and Mechatronics, Vol.22 No.2, pp. 230-238, 2010.
[10] D. Katz, J. Kenney, and O. Brock, “How Can Robots Succeed in Unstructured Environments”, *In Robotics Science and Systems (RSS) Workshop on Robot Manipulation*, 2008.
[11] K. Nagata et al., “Picking up an Indicated Object in a Complex Environment”, *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp.2109-2116, 2010.
[12] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, “Push Planning for Object Placement on Cluttered Table Surfaces”, *Proc. of IEEE/RSJ int. Conf. on Intelligent Robots and Systems*, 2011.
[13] M.J. Schuster, J. Okerman, H. Nguyen, J.M. Rehg, and C.C. Kemp, “Perceiving Clutter and Surfaces for Object Placement in Indoor Environments”, *Proc. of IEEE-RAS Int. Conf. on Humanoid Robots*, 2010.
[14] E. Klingbeil, A. Saxena, and A. Y. Ng, “Learning to Open New Doors”, *In Robotics Science and Systems (RSS) Workshop on Robot Manipulation*, 2008.
[15] N. Bergstrom, J. Bohg, and D. Kragic, “Integration of Visual Cues for Robotic Grasping”, *Proc. 7th Int. Conf. on Computer Vision Systems*, 2009.
[16] M. Garland, A. Willmott, and P. S. Heckbert, ”Hierarchical Face Clustering on Polygonal Surfaces”, *Proc. of ACM Symposium on Interactive 3D Graphics*, 2001.
[17] T. Tsuji, H. Zha, T. Hasegawa, and R. Kurazume, ”Levels of Detail Control Based on Correlation Analysis Between Surface Position and Direction”, *Proc. of Int. Conf. on Pattern Recognition*, 2004.
[18] PCL-Point Cloud Library, http://pointclouds.org/
[19] Princeton Shape Benchmark, http://shape.cs.princeton.edu/benchmark/
[20] graspPlugin for Choreonoid, http://www.choreonoid.org/GraspPlugin/
[21] QuickHull, http://qhull.org/