2D Caging Manipulation by Robots and Walls

Ryo YOKOI
Maeda Lab, Dept. of Mechanical Engineering,
Div. of Systems Integration,
Graduate School of Engineering,
Yokohama National University
79-5 Tokiwadai, Hodogaya-ku, Yokohama,
240-8501 Japan
Email: yokoi@iir.me.ynu.ac.jp

Tatsuya KOBAYASHI
Komatsu Ltd.

Yusuke MAEDA
Dept. of System Design, Div. of Systems Research,
Faculty of Engineering,
Yokohama National University
Email: maeda@ynu.ac.jp

***Abstract*—In this paper, we propose a new robotic manipulation: caging manipulation by robots and walls. Caging is a method to make an object inescapable from a closed region by rigid bodies. Previous studies use only robots for caging, while we use walls as well. First, we formulate 2D caging with the environment such as walls. Secondly, we derive the caging manipulability condition. The manipulability problem is trivial in robot-only caging, but crucial in our cases. Finally, we present a method to plan 2D caging manipulation of a circular object by circular robots and straight walls.**

I. Introduction

In conventional robotic manipulation, form- or force-closure grasping is used to constrain movements of objects. Caging (or capturing) is another method to constrain objects for manipulation [1]; in caging, an object is inescapable from a closed region without penetrating robot bodies. The state where caging is achieved is termed *object closure* [2][3].

Caging has several advantages over grasping. Mechanical analysis is not necessary because the object is constrained geometrically. That is to say, position-controlled robots can be used for manipulation. Moreover, we can tolerate positional control errors of robots to some extent by considering a margin for caging. Robots with few degrees of freedom can be used for caging.

Rimon and Blake proposed a 2D caging method to constrain concave objects by using two-fingered hands with disc-shaped fingertips [1]. Wang and Kumar proposed a 2D caging algorithm by multiple mobile robots [2]. Pereira et al. also dealt with 2D caging by multiple mobile robots [3]. Some caging algorithms were also proposed for 2D caging by disc-shaped fingertips [4][5], nD caging by point fingertips [6] and 3D caging by link mechanisms [7]. These studies basically discussed robot-only caging.

On the other hand, we can consider caging with the environments; that is, caging by not only robots but also walls. Using the environment brings us some advantages. First, the number of robots required for caging can be fewer than robot-only caging (Fig.1). Secondly, it may be possible to cage objects in the narrow environment where robot-only caging is infeasible.

However, caging with the environment has a difficulty in the manipulability. In robot-only caging, once an object is caged, robots can manipulate it simply by translation keeping their relative positions. Thus the manipulability problem is trivial. However, it is not the case for caging by robots and walls; translation may not be enough to manipulate objects (Fig.2), and robot formation may has to be changed during manipulation (Fig.3). Hence we have to judge whether caging manipulation by robots and walls is possible or not.

In this paper, we discuss 2D caging manipulation by robots

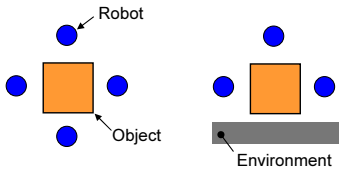

Fig. 1. Robot-only Caging and Caging by Robots and Walls

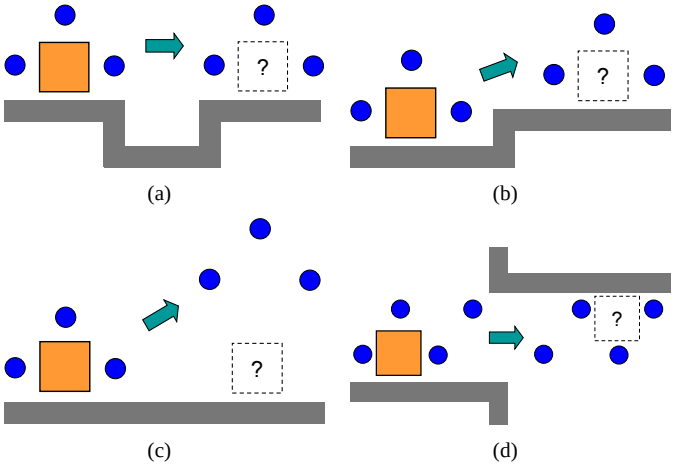

Fig. 2. Some Situations Where Manipulation Is Impossible by Translation

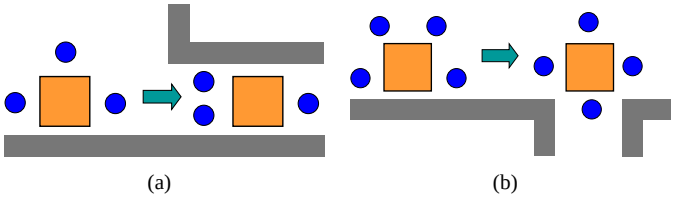

Fig. 3. Some Situations Where Changing Formation Is Required for Manipulation

and walls. First, we formally define caging with the environment. Then we formulate caging manipulability. Finally, we propose a method to plan robot motions for caging manipulation with the environment. The method is based on the result of caging manipulability analysis. Some planning results are also shown.

II. Formulation of Caging with Environment

Let us formulate caging by robots and walls. We assume that the robots, the object and the environment such as walls are all rigid.

We use the following symbols:

- n: number of robots.
- $\mathcal{C}$: configuration space of the object.
- $\mathcal{A}_{\rm obj}$: region occupied by the object in the real space.
- $\mathcal{A}_i$: region occupied by the ith robot in the real space ($i = 1, \ldots, n$).
- $\mathcal{E}$: region occupied by the environment in the real space.
- $\boldsymbol{q}_{\rm obj}$: configuration of the object.
- $\boldsymbol{q}_i$: configuration of the ith robot ($i = 1, \ldots, n$).
- $\boldsymbol{q}_{\rm rob} = [\boldsymbol{q}_1^T, \ldots, \boldsymbol{q}_n^T]^T$.

Now we define object closure for caging with the environment. This is a straightforward extension of object closure for robot-only caging presented in [2].

First, we define the configuration obstacle of ith robot or C-Closure Object, $\mathcal{C}_{{\rm cls}\_i}$, as follows:

$$\mathcal{C}_{{\rm cls}\_i} = \{\boldsymbol{q}_{\rm obj} \in \mathcal{C} | \mathcal{A}_{\rm obj}(\boldsymbol{q}_{\rm obj}) \cap \mathcal{A}_i(\boldsymbol{q}_i) \neq \emptyset\}. \quad (1)$$

Similarly, the configuration obstacle of the environment, $\mathcal{C}_{\rm cls\_env}$, is defined as follows:

$$\mathcal{C}_{\rm cls\_env} = \{\boldsymbol{q}_{\rm obj} \in \mathcal{C} | \mathcal{A}_{\rm obj}(\boldsymbol{q}_{\rm obj}) \cap \mathcal{E} \neq \emptyset\}. \quad (2)$$

The total configuration obstacle region or C-Closure Object Region, $\mathcal{C}_{\rm cls}$, is given by the union of the configuration obstacles as follows:

$$\mathcal{C}_{\rm cls} = \bigcup_{i=1}^{n} \mathcal{C}_{{\rm cls}\_i} \cup \mathcal{C}_{\rm cls\_env}. \quad (3)$$

The free space of the obstacle, $\mathcal{C}_{\rm free}$, where the object is free from interferences with the robots and the environment, is written as follows:

$$\mathcal{C}_{\rm free} = \mathcal{C} \setminus \mathcal{C}_{\rm cls} = \left(\mathcal{C} \setminus \bigcup_{i=1}^{n} \mathcal{C}_{{\rm cls}\_i} \right) \setminus \mathcal{C}_{\rm cls\_env}. \quad (4)$$

Let $\boldsymbol{q}_{\rm obj}$ be a free configuration of the object. Then we define a subset of $\mathcal{C}_{\rm free}$, $\mathcal{C}_{\rm free\_obj}$, as follows:

$$\mathcal{C}_{\rm free\_obj} = \{\boldsymbol{q} \in \mathcal{C}_{\rm free} | {\rm connected}(\boldsymbol{q}, \boldsymbol{q}_{\rm obj})\}. \quad (5)$$

$\mathcal{C}_{\rm free\_obj}$ is the maximal connected subset of $\mathcal{C}_{\rm free}$ that contains $\boldsymbol{q}_{\rm obj}$.

Similarly, let $\boldsymbol{q}_{\rm inf}$ be a point at infinity in $\mathcal{C}$. Then we define another subset of $\mathcal{C}_{\rm free}$, $\mathcal{C}_{\rm free\_inf}$, as follows:

$$\mathcal{C}_{\rm free\_inf} = \{\boldsymbol{q} \in \mathcal{C}_{\rm free} | {\rm connected}(\boldsymbol{q}, \boldsymbol{q}_{\rm inf})\}. \quad (6)$$

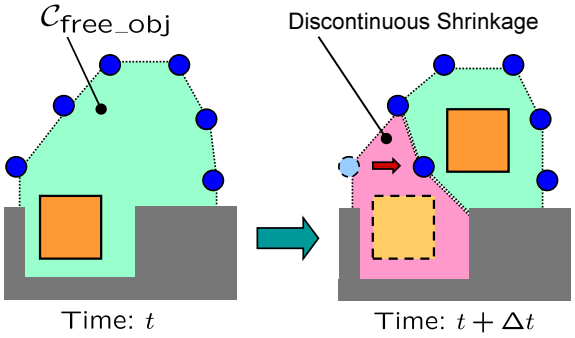

Fig. 4. Discontinuous Shrinkage of $\mathcal{C}_{\rm free\_obj}$

The object is in object closure if and only if there is no feasible path from $\boldsymbol{q}_{\rm obj}$ to $\boldsymbol{q}_{\rm inf}$. Thus the following conditions must be satisfied in object closure for caging with the environment:

$$\begin{cases} \mathcal{C}_{\rm free\_obj} \neq \emptyset \\ \mathcal{C}_{\rm free\_obj} \cap \mathcal{C}_{\rm free\_inf} = \emptyset. \end{cases} \quad (7)$$

III. Caging Manipulability

In conventional robot-only caging, once object closure is achieved, the robots can manipulate the object simply by translation keeping their relative positions. Thus the problem of the caging manipulability is trivial. On the other hand, in caging with the environment, the manipulability is not trivial as mentioned in Section I. How can we test the manipulability in caging by robots and walls?

We focus attention on the change of $\mathcal{C}_{\rm free\_obj}$ between time t and time $t + \Delta t$. $\mathcal{C}_{\rm free\_obj}$ may shrink discontinuously even if the robots move continuously (Fig.4). In caging, the configuration of the object can be arbitrary in $\mathcal{C}_{\rm free\_obj}$. Accordingly, discontinuous shrinkage of $\mathcal{C}_{\rm free\_obj}$ may mean discontinuous movement of the object, which is physically infeasible. Thus such discontinuous shrinkage is not allowed in caging manipulation. Note that discontinuous expansion of $\mathcal{C}_{\rm free\_obj}$ is allowed.

Therefore the following condition must be satisfied for caging manipulation:

$$\lim_{\Delta t \to +0} (\mathcal{C}_{\rm free\_obj}(t) \cap \mathcal{C}_{\rm free\_obj}(t + \Delta t)) = \mathcal{C}_{\rm free\_obj}(t). \quad (8)$$

This is the caging manipulability condition and must be satisfied in addition to the object closure condition (7). Note that this condition is satisfied implicitly in robot-only caging by continuous translation of robots.

$\mathcal{C}_{\rm free\_obj}$ may split into two or more subsets between time t and time $t + \Delta t$. In such cases, we cannot determine which subset contains the object. However, caging manipulation may be possible because the subsets may join together later. Thus the caging manipulability condition in such special cases can be written as follows:

$$\lim_{\Delta t \to +0} \left(\mathcal{C}_{\rm free\_obj}(t) \cap \left(\bigcup_i \mathcal{C}_{{\rm free\_obj}\_i}(t + \Delta t) \right) \right) = \mathcal{C}_{\rm free\_obj}(t), \quad (9)$$

where $\mathcal{C}_{\text{free\_obj\_}i}$ is the ith split subset. For simplicity, however, we do not consider such splitting cases from this point.

Here we used the term "manipulability"; however, even if (8) is satisfied, the robot motion may not be able to move the object at the instant. (For example, note that the robots can push the object but cannot pull it.) Such a robot motion does not move the object at the instant but may be necessary in the total caging manipulation (e.g. formation change).

IV. Planning of Caging Manipulation by Robots and Walls

A. Overview of Planning

The planning problem for caging manipulation is to find a motion of the robots that transfers the object to a goal area. Here we present a motion planning algorithm for caging manipulation using the environment such as walls. This algorithm uses the object closure condition (7) and the caging manipulability condition (8) for planning.

The following assumptions are made for simplicity:

- The object has a circular shape whose radius is R_{obj}.
- All the robots have a congruent circular shape whose radius is R_{rob}.
- All robots are holonomic.
- The ith robot is at $\boldsymbol{q}_{\text{ini\_}i}$ in the initial state. $\boldsymbol{q}_{\text{rob\_ini}} = [\boldsymbol{q}_{\text{ini\_}1}^T, \ldots, \boldsymbol{q}_{\text{ini\_}n}^T]^T$.
- The object is at $\boldsymbol{q}_{\text{obj\_ini}}$ and in object closure in the initial state.
- The total system is a discrete time system with a time interval Δt.

We do not consider the orientation of the object and the robots because they are circular. The discrete-time assumption is required for object closure test and caging manipulability test, which are described later.

Since the object configuration cannot be determined uniquely in caging, the robots cannot transfer the object to a unique configuration. Accordingly, the goal condition for planning of caging manipulation must be different from that for conventional manipulation planning.

Let $\boldsymbol{q}_{\text{goal}}$ be the representative goal configuration of the object. $\boldsymbol{q}_{\text{goal}} \in \mathcal{C}_{\text{free\_obj}}$ is usually not enough for the goal condition, because $\mathcal{C}_{\text{free\_obj}}$ may be very large and the object may be far from $\boldsymbol{q}_{\text{goal}}$ even if $\boldsymbol{q}_{\text{goal}} \in \mathcal{C}_{\text{free\_obj}}$. Thus we define the goal condition as follows so that the object will be transferred close to $\boldsymbol{q}_{\text{goal}}$:

$$||\boldsymbol{q}_{\text{obj}} - \boldsymbol{q}_{\text{goal}}|| \leq R_{\text{goal}} \text{ for } \forall \boldsymbol{q}_{\text{obj}} \in \mathcal{C}_{\text{free\_obj}}. \quad (10)$$

In other words, the circle of radius R_{goal} centered on $\boldsymbol{q}_{\text{goal}}$ contains $\mathcal{C}_{\text{free\_obj}}$ (Fig.5).

B. Object Closure Test

We must test whether the object is in object closure or not based on (7) in planning.

We adopt a grid-based approximate representation for the configuration space of the object, because its exact representation can be very complex. We use the center of the object as its representative point, and possible locations of the center can be represented by 2D grid points.

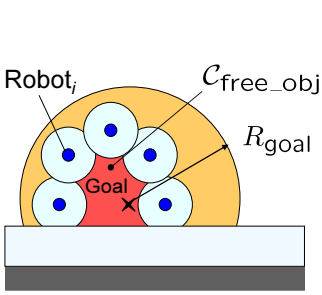

Fig. 5. Goal Condition

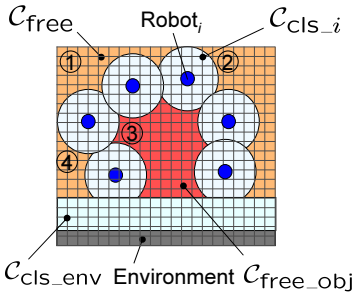

Fig. 6. Labeling Process

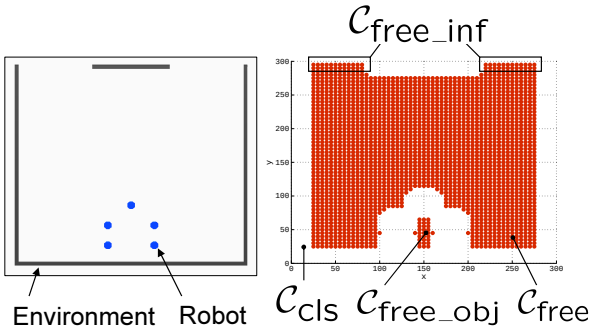

Fig. 7. Example of $\mathcal{C}_{\text{free\_inf}}$

Grid points where the object interferes with the robots or the environment are in $\mathcal{C}_{\text{cls}}$. Thus by removing such grid points we can obtain an approximate representation of $\mathcal{C}_{\text{free}}$.

Since $\mathcal{C}_{\text{free}}$ may be split into some disconnected regions, we need to identify $\mathcal{C}_{\text{free\_obj}}$, in which the object exists. We cannot determine the configuration of the object uniquely in caging, thus a little bit complicated procedure is required.

First, we perform connected component labeling to find connected regions in $\mathcal{C}_{\text{free}}$ (Fig.6). $\mathcal{C}_{\text{free\_obj}}(0)$ is the connected region that includes $\boldsymbol{q}_{\text{obj\_ini}}$. $\mathcal{C}_{\text{free\_obj}}(t + \Delta t)$ $(t \geq 0)$ is the connected region that has the maximum intersection with $\mathcal{C}_{\text{free\_obj}}(t)$. There may be two or more regions that have the maximum intersection; however, such situations can be safely omitted because the caging manipulability condition (8) is not satisfied.

Then we define $\mathcal{C}_{\text{free\_inf}}$ as the set of grid points at the border, instead of a point at infinity. In the case of Fig.7(a), for example, $\mathcal{C}_{\text{free\_inf}}$ is composed of grid points at the top left and the top right as shown in Fig.7(b). If $\mathcal{C}_{\text{free\_obj}}$ is not empty and does not intersect with $\mathcal{C}_{\text{free\_inf}}$, we judge that (7) is satisfied and the object is in object closure. Otherwise, the object is not caged.

C. Caging Manipulability Test

We must also test whether a robot motion is a valid caging manipulation or not based on (8) in planning.

Due to our discrete space-time representation, (8) can be

approximated by the following inequality:

$$\frac{n(\mathcal{C}_{\text{free\_obj}}(t) \setminus \mathcal{C}_{\text{free\_obj}}(t+\Delta t))}{n(\mathcal{C}_{\text{free\_obj}}(t))} < a, \tag{11}$$

where $n(\mathcal{C})$ is the number of grid points included in the region $\mathcal{C}$; a is a threshold for the continuity of $\mathcal{C}_{\text{free\_obj}}$ and $a < 1$. If (11) is satisfied, the shrinkage of $\mathcal{C}_{\text{free\_obj}}$ is very small, if any.

However, (11) is too harsh in some cases; if $a \times n(\mathcal{C}_{\text{free\_obj}}(t)) < 1$, the number of grid points $n(\mathcal{C}_{\text{free\_obj}}(t) \setminus \mathcal{C}_{\text{free\_obj}}(t+\Delta t))$ must be zero. Therefore, we allow the shrinkage of $\mathcal{C}_{\text{free\_obj}}$ up to one grid point. This condition is written as follows:

$$n(\mathcal{C}_{\text{free\_obj}}(t) \setminus \mathcal{C}_{\text{free\_obj}}(t+\Delta t)) \leq 1. \tag{12}$$

If (11) or (12) is satisfied, we regard the corresponding robot motion as valid caging manipulation.

D. Motion Planning of Robots

We adopt RRT (Rapidly-exploring Random Trees) [9] for motion planning. The procedure of motion planning of the robots based on RRT is as follows:

1) Add $\boldsymbol{q}_{\text{rob\_ini}}$ to the RRT.
2) Sample a configuration of the robots: $\boldsymbol{q}_{\text{sample}}$, and find its nearest neighbor in the RRT: $\boldsymbol{q}_{\text{nearest}}$.
3) Consider a branch of length Δl from $\boldsymbol{q}_{\text{nearest}}$ to $\boldsymbol{q}_{\text{new}}$ in the direction of $\boldsymbol{q}_{\text{sample}}$.
4) If there are collisions among the robots and the environment at $\boldsymbol{q}_{\text{new}}$, discard $\boldsymbol{q}_{\text{new}}$ and go back to step 2).
5) If $\boldsymbol{q}_{\text{new}}$ does not pass the object closure test, discard $\boldsymbol{q}_{\text{new}}$ and go back to step 2).
6) If $\boldsymbol{q}_{\text{new}}$ does not pass the caging manipulability test, discard $\boldsymbol{q}_{\text{new}}$ and go back to step 2).
7) Add the branch from $\boldsymbol{q}_{\text{nearest}}$ to $\boldsymbol{q}_{\text{new}}$ to the RRT.
8) Repeat steps from 2) to 7) until $\boldsymbol{q}_{\text{new}}$ satisfies the goal condition (10).

If the goal condition is satisfied, a path from the initial configuration to the final configuration in the RRT can be obtained. The path is the planned robot motion for caging manipulation with the environment. In step 3), $\boldsymbol{q}_{\text{new}}$ is calculated as follows:

$$\boldsymbol{q}_{\text{new}} = \boldsymbol{q}_{\text{nearest}} + \Delta l \frac{\boldsymbol{q}_{\text{sample}} - \boldsymbol{q}_{\text{nearest}}}{||\boldsymbol{q}_{\text{sample}} - \boldsymbol{q}_{\text{nearest}}||}. \tag{13}$$

The above procedure spends much time for planning if $\boldsymbol{q}_{\text{sample}}$ is sampled completely at random. Therefore, we add some heuristics to bias the sampling to accelerate planning. In step 2) of the above procedure, we determine $\boldsymbol{q}_{\text{sample}}$ and $\boldsymbol{q}_{\text{nearest}}$ by the following strategies:

a) Random Motion for Each Robot: We just determine $\boldsymbol{q}_{\text{sample}}$ at random and find its nearest neighbor in the RRT: $\boldsymbol{q}_{\text{nearest}}$.

TABLE I
PARAMETERS IN PLANNING

Planning Region	600×600	Grid Interval	1
R_{rob}	10	a	0.01
R_{obj}	40	P_a	0.8
R_{goal}	100	P_b	0.1
Δl	0.1	P_c	0.1

b) Random Translation: We consider translation of the robots by the following procedure:

1) Sample a random object configuration: $\boldsymbol{q}_{\text{sample\_obj}}$.
2) Find its nearest neighbor in the RRT: $\boldsymbol{q}_{\text{nearest}}$. Here we use the following distance function between a robot configuration and an object configuration:

$$d(\boldsymbol{q}_{\text{rob}}, \boldsymbol{q}_{\text{obj}}) = \sqrt{\sum_{i=1}^{n} ||\boldsymbol{q}_i - \boldsymbol{q}_{\text{obj}}||^2}. \tag{14}$$

3) Calculate the centroid of the robots that contribute to object closure: $\boldsymbol{q}_{\text{nearest\_cent}}$ and a vector: $\boldsymbol{v} = \boldsymbol{q}_{\text{sample\_obj}} - \boldsymbol{q}_{\text{nearest\_cent}}$.
4) Determine $\boldsymbol{q}_{\text{sample}}$ so that $\boldsymbol{q}_{\text{sample\_}i} = \boldsymbol{q}_{\text{nearest\_}i} + \boldsymbol{v}$ for the robots that contribute to object closure. $\boldsymbol{q}_{\text{sample\_}i}$ for the robots that do not contribute to object closure is determined at random.

As a result, the robots that contribute to object closure are translated in a random direction. The robots that do not contribute to object closure are moved in a random direction.

c) Translation to Goal: We consider translation of the robots to the goal by the following procedure:

1) Sample $\boldsymbol{q}_{\text{goal}}$ as an object configuration.
2) Find its nearest neighbor in the RRT: $\boldsymbol{q}_{\text{nearest}}$. Here we use (14) as the distance function.
3) Calculate the centroid of the robots that contribute to object closure: $\boldsymbol{q}_{\text{nearest\_cent}}$ and a vector: $\boldsymbol{v} = \boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{nearest\_cent}}$.
4) Determine $\boldsymbol{q}_{\text{sample}}$ so that $\boldsymbol{q}_{\text{sample\_}i} = \boldsymbol{q}_{\text{nearest\_}i} + \boldsymbol{v}$ for the robots that contribute to object closure. $\boldsymbol{q}_{\text{sample\_}i}$ for the robots that do not contribute to object closure is determined at random.

As a result, the robots that contribute to object closure are translated in the goal direction. The robots that do not contribute to object closure are moved in a random direction.

We select the above strategies with a probability of P_a, P_b and P_c, respectively ($P_a + P_b + P_c = 1$). These heuristics will increase the probability of searching toward the goal and reduce the planning time.

E. Results of Motion Planning

We implemented the above procedure based on MSL (Motion Strategy Library) [8]. The implemented planner was tested on a Linux PC with Core2Quad Q9450 CPU at 2.66 (GHz). The parameters used in motion planning are listed in TABLE I.

Even if our planner successfully finds a robot motion for caging manipulation with the environment, the object configuration during manipulation cannot be determined by definition of caging; that is, we cannot visualize the object motion in planned manipulation uniquely.

However, it is inconvenient that visual verification of planned manipulation is impossible, thus we simulate planned results on a dynamics simulator, ODE (Open Dynamics Engine) [10]. Such simulation does not prove the correctness of the planner but provides useful information on its validity. The robots in the simulator are in PID control to follow the planned paths. We assume that the robots have almost infinite mass so that their motion is not affected by the object. (Note that caging manipulation is completely based on geometry.)

1) Caging Manipulation by Three Robots along a Wall: We show a planning result of caging manipulation by three robots. The robots are initially at the lower left in Fig.8 and the goal is at the lower right. Fig.9 shows constructed RRT of the three robots and Fig.10 shows dynamic simulation of planned caging manipulation on ODE. Our planner successfully found a robot motion for caging manipulation along a wall, while the planned motion is not well-optimized. The computation time for planning was 604 CPU seconds.

2) Caging Manipulation by Two Robots in a Corridor: We show a planning result of caging manipulation by two robots. The robots are initially at the left of a corridor as shown in Fig.11 and the goal is at the right. The corridor is narrow and therefore only two robots can cage the object. Fig.12 shows constructed RRT of the right robot and Fig.13 shows dynamic simulation of planned caging manipulation on ODE. Our planner successfully found a robot motion for caging manipulation by sandwiching the object. The computation time for planning was 465 CPU seconds.

3) Caging Manipulation by Two Robots in an L-shaped Corridor: We show another planning result of caging manipulation by two robots. The robots are initially at the lower left of an L-shaped corridor as shown in Fig.14 and the goal is at the upper right. Fig.15 shows constructed RRT of the robot initially at right, and Fig.16 shows dynamic simulation of planned caging manipulation on ODE. Our planner successfully found a robot motion for caging manipulation, while the planned motion is not well-optimized. The computation time for planning was 4606 CPU seconds.

V. Conclusion

In this study, we presented caging manipulation with the environment such as walls. Such manipulation has several advantages over conventional robot-only caging. We formally defined caging with the environment and derived a condition for the caging manipulability. We also presented an RRT-based planner for caging manipulation with the environment. The planner worked successfully in several cases.

Our planner is still under development and can stand further improvement in many aspects. The computation efficiency is one of the most important issue. The shape limitation of objects and robots should be relaxed. Planned paths of robots should be smoothed to omit unnecessary motions. We will also address a design of part feeder based on the idea of caging manipulation with the environment.

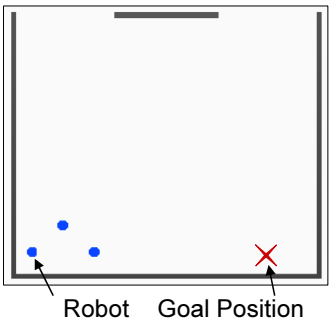

Fig. 8. Initial State of Caging Manipulation

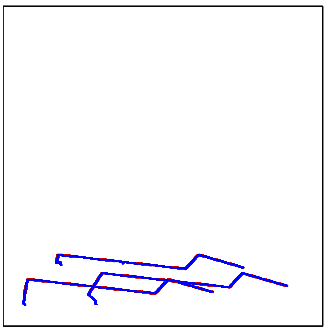

Fig. 9. RRT of Three Robots

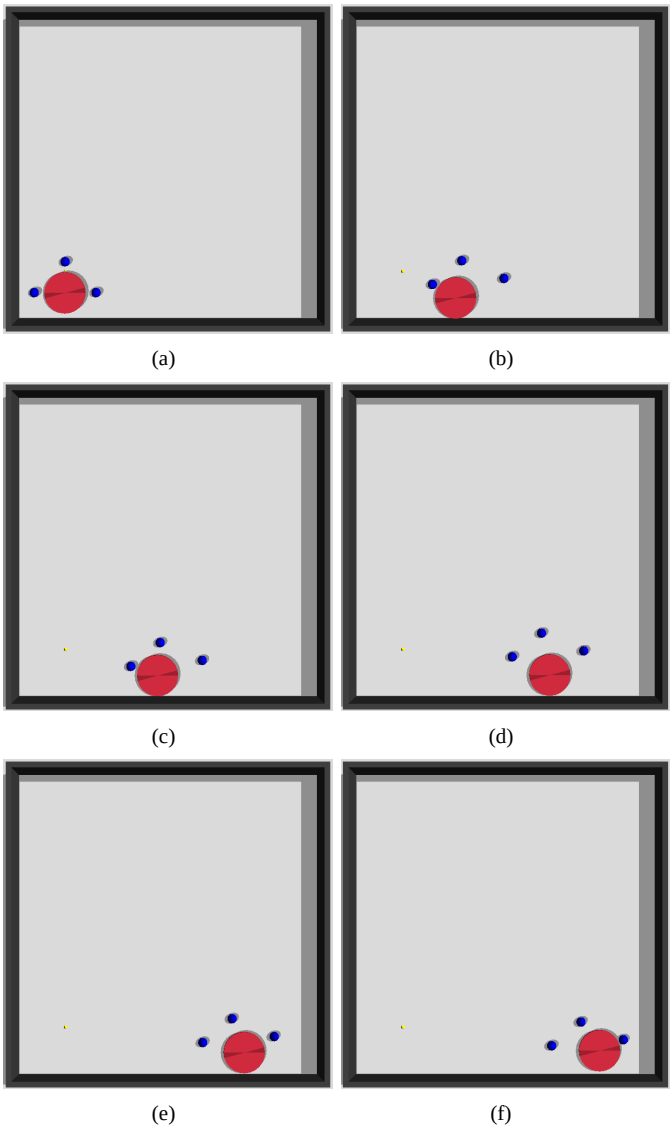

Fig. 10. Planned Caging Manipulation along a Wall

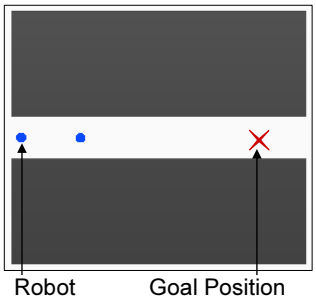

Fig. 11. Initial State of Caging Manipulation

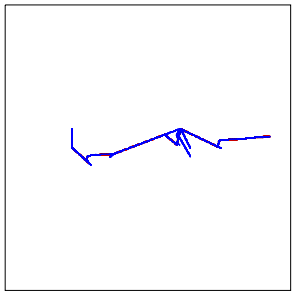

Fig. 12. RRT of The Right Robot

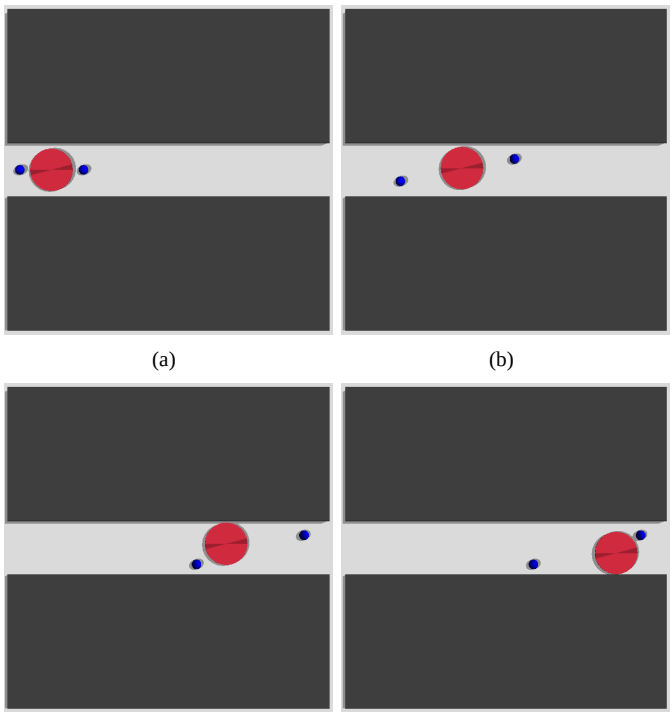

Fig. 13. Planned Caging Manipulation in a Corridor

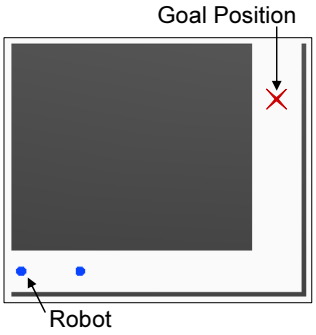

Fig. 14. Initial State of Caging Manipulation

Fig. 15. RRT of The Right Robot

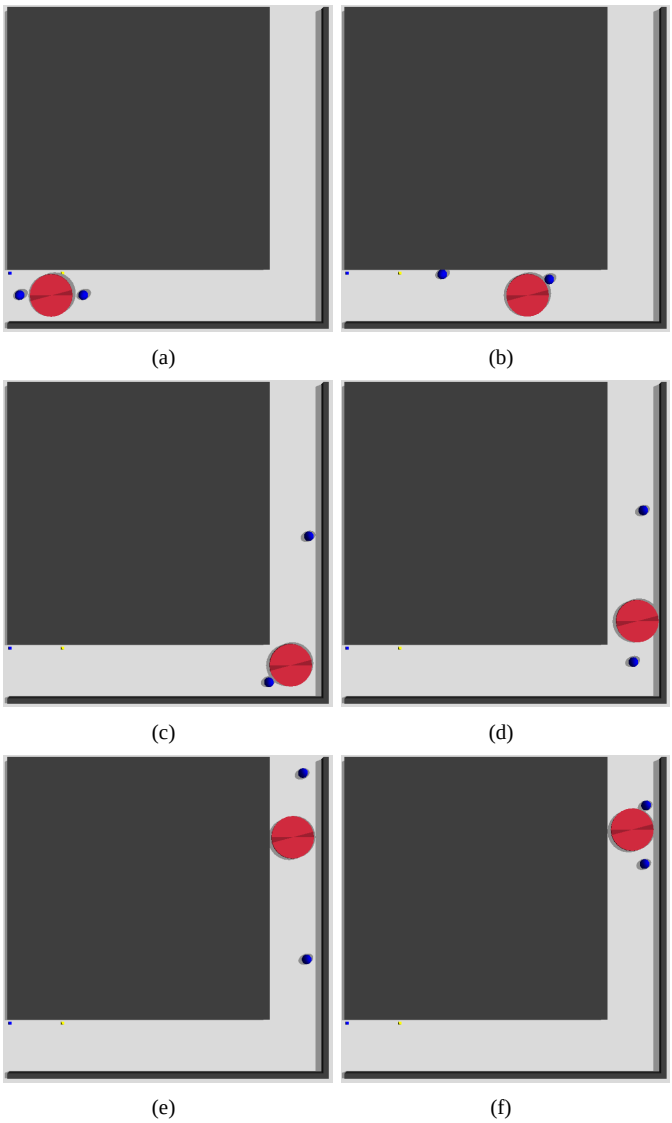

Fig. 16. Planned Caging Manipulation in an L-shaped Corridor

REFERENCES

[1] E. Rimon and A. Blake: “Caging Planar Bodies by One-Parameter Two-Fingered Gripping Systems,” Int. J. of Robotics Research, Vol. 18, No. 3, pp. 299–318, 1999.
[2] Z. Wang and V. Kumar: “Object Closure and Manipulation by Multiple Cooperating Mobile Robots,” in Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 394–399, 2002.
[3] G. A. S. Pereira, M. F. M. Campos and V. Kumar: “Decentralized Algorithms for Multi-Robot Manipulation via Caging,” Int. J. of Robotics Research, Vol. 23, No. 7–8, pp. 783–795, 2004.
[4] J. Erickson, S. Thite, F. Rothganger and J. Ponce: “Capturing a Convex Object With Three Discs,” IEEE Trans. on Robotics, Vol. 23, No. 6, pp.1133–1140, 2007.
[5] M. Vahedi and A. F. van der Stappen: “Caging Polygons with Two and Three Fingers,” Int. J. of Robotics Research, Vol. 27, No. 11–12, pp. 1308–1324, 2008.
[6] P. Pipattanasomporn, P. Vongmasa and A. Sudsang: “Caging Rigid Polytopes via Finger Dispersion Control,” in Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 1181–1186, 2008.
[7] S. Makita and Y. Maeda: “3D Multifingered Caging: Basic Formulation and Planning,” in Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 2697–2702, 2008.
[8] “Motion Strategy Library,” http://msl.cs.uiuc.edu/msl/.
[9] S. M. LaValle and J. J. Kuffner: “Rapidly-exploring Random Trees: Progress and Prospects,” B. R. Donald, K. M. Lynch and D. Rus eds., Algorithmic and Computational Robotics: New Directions, A. K. Peters, pp. 293–308, 2001.
[10] “Open Dynamics Engine,” http://www.ode.org/.