Motion Planning for 3D Multifingered Caging with Object Recognition using AR Picture Markers

Satoshi Makita, Kensuke Okita and Yusuke Maeda

Abstract— **Caging is a method to capture an object geometrically by position-controlled robots without any force and tactile sensors. Many previous researches focused on caging constraints of objects, and those on planning are few. In this paper, we present a motion planner for caging by a multifingered hand and a manipulator to produce whole motion which includes approaching to a target object and capturing it without any collisions. We derive sufficient conditions required for the caging tasks about three caging patterns. Since the planner requires the object properties including the position and orientation of the object, we adopt an object recognition using AR picture markers. We apply the proposed method to caging about four target objects: a cylinder, a ring, a mug and a dumbbell. Some experimental results shows that each motion are successfully planned, and executed by the arm/hand system.**

I. Introduction

Robotic manipulations are ofter achieved with grasping, which is typically performed by a force-controlled robot hand, and then contact points between the robot and a target object, and contact forces applied to the object have to be determined [1]. Thus force sensors and/or tactile sensors are often necessary, and force or torque control of robots are also required.

On the other hand, Caging can be performed by a position-controlled robot hand, and then an object is geometrically constrained in the *cage* composed of the hand [2]. Since we only determine the configuration of the hand to constrain the object, any force sensors and force control are not required.

Moreover, there is room for object to move in the cage. The room plays as margins of caging constraint to allow some errors caused in position control, modeling and estimation of link parameters. These advantages contribute to facilitate execution of manipulation with actual robots. For example, caging grasps, which are called in [3] as grasp sets without constraints through contact forces, were presented to open a door by a humanoid robot. Caging could provide the robot more reachability to accomplish the manipulation task.

We derived sufficient conditions for caging about three objects (Fig. 1, 2, 3) [4]. The sufficient conditions ensure that the robot hand can form a cage to make a target object not be able to escape from it. In addition, we obtain the required configuration of the robot hand for caging by using a path planner based on RRT (Rapidly exploring Random Trees) [5]. However, it can produce only required joint angles of the hand.

Many of studies on caging focuses on the formation of robots: caging a concave object by two fingertips in 2D scenes [2], [6]; caging a convex polygon by more than three disks in 2D [7], [8]; caging a polyhedra by pointed fingers in n-dimension [9]; caging by a multifingered hand in 3D [4]. Motion planning for caging is, however, important for execution. Wang et al. proposed 2D caging by multiple mobile robots including approaching phase [10]. Diankov et al. demonstrated manipulating the door with caging grasps by a multifingered hand [3], but mathematical conditions of caging constraints are not declared.

In this paper, we improve the path planner presented in our previous work [4], which can produce only finger locations for caging, to produce motions for caging by a robotic hand and a manipulator. The motions include both the hand approaching to a target object from an initial state and capturing the object. Thus we additionally derive sufficient conditions to examine whether the hand approaches enough to capture. Moreover, we propose a method of object recognition with AR picture markers and ARToolKitPlus [11], [12], because the planner requires object properties: the shape, the size and the position and orientation of the object, to plan a path of robot motion. The markers are useful to identify each object, and the posture of the marker can be measured with the libraries in ARToolKitPlus.

With the sufficient conditions and the improved planner, we produce robot motions for caging about two objects: a ring-like object and a dumbbell-like object. These caging can be achieved by two-fingered hand. Although we presented caging for both a sphere and a disk [4], we do not deal with them in this paper. It is because a robot hand cannot surround these objects without contacts with the obstacles such as floors, and we cannot produce robot motions for these caging by using our current planner.

II. Conditions for 3D Multifingered Caging

A. Classification of 3D Multifingered Caging

First, we name 3D multifingered caging to classify various caging into three types by focusing on characters of objects' shapes and constraint states.

1) Envelope-type Caging: In such cases of caging a sphere (Fig. 1) or a disk (Fig. 2), finger bodies and a palm of a robot hand surround the object and make it inescapable.

2) Ring-type Caging: In such cases of caging a ring-like object, a robot hand constrains the object inserting its fingers

S. Makita is with Dept. of Control Engineering, Sasebo National College of Technology, 1-1 Okishinmachi, Sasebo, Japan. makita@sasebo.ac.jp

K. Okita is with Canon Inc. b0541033@yahoo.co.jp

Y. Maeda is with Dept. of Systems Design, Div. of Systems Research, Faculty of Engineering, Yokohama National University, 79-5 Tokiwadai, Hodogaya-ku, Yokohama, Japan. maeda@ynu.ac.jp

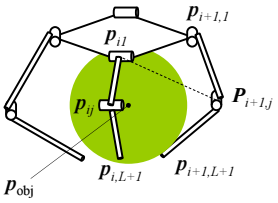


Fig. 1. Envelope-type caging (Caging a sphere)

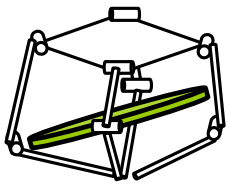


Fig. 2. Envelope-type caging (Caging a disk)

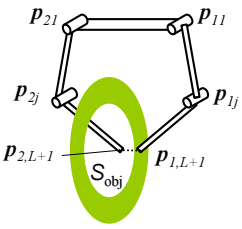


Fig. 3. Ring-type caging

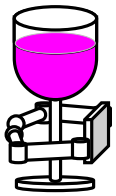


Fig. 4. Waist-type caging (Caging a wineglass)

into a hollow of the object (Fig. 3). Ring-type caging can be considered for objects that has ring parts such as mugs.

3) Waist-type Caging: In such cases of caging an object that has a constricted part such as a dumbbell (Fig. 4, 5), a robot hand constrains the object winding its fingers around the constricted part of the object.

B. Sufficient Conditions for Caging

If all the following conditions are satisfied, caging an object by a robot hand can be accomplished.

1) **Cage-formed conditions** The robot hand constructs the caged region from which the object cannot escape.
2) **In-cage conditions** The object is present in the caged region formed by the robot.
3) **Collision-free conditions** The robot hand has no collision with the object and any obstacles.

In this paper, we discuss cage-formed conditions and in-cage conditions. Especially, in-cage conditions are necessary for planning robot motion of approaching to the target object. We derive in-cage conditions of each caging types classified above, and one series of cage-formed conditions for caging a dumbbell-like object, which is not appeared in [4].

As [4], collision-free tests among the hand, the object and any obstacles can be checked with PQP — a Proximity Query Package [13].

C. Assumptions and Notations

As [4], we assume a symmetric robot hand as follows:

- The hand has N fingers, and each finger has $\bar{L}$ joints.
- All the joints are revolute, and their bodies have no volume, that is, they can be approximated by points.
- The j-th body of the i-th finger can be approximated by a line segments with length l_j.
- The palm of the robot hand is a regular plane with N vertices. When $N = 2$, it is assumed to be a square.
- Each finger is attached to each vertex of the palm.

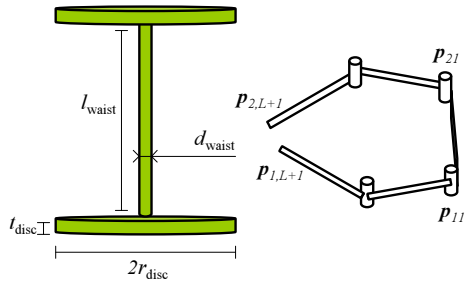


Fig. 5. Caging a dumbbell-like object (Waist-type caging)

- Each finger can move only in the plane passing through both the vertical center axis of the palm and each vertex of the palm.

The movements of the fingers have rotational symmetries through $360/N$ degrees about the vertical center axis of the palm. Then, we define a vector of joint variables of every finger as follows:

$$\bar{\boldsymbol{\theta}} := [\theta_1, \theta_2, \ldots, \theta_{\bar{L}}]^T, \tag{1}$$

where θ_j is a joint variable of the j-th joint of each finger.

D. Cage-formed Conditions

In this section, we derive *cage-formed conditions*, which ensures that a robot hand constructed a caged region and makes an object inescapable from it, for caging a dumbbell-like object. Caging a dumbbell-like object is a kind of *Waist-type caging* (Sec. II-A.3). Note that the cage-formed conditions for some other classified types of caging: envelope-type caging and ring-type caging have already been derived in [4].

1) Caging a dumbbell-like object: Let us consider caging a dumbbell-like object by a robot hand with two fingers ($N = 2$) as Fig. 5. The object is composed of a cylinder with two circular plates (disks) attached on each base, and their parameters are denoted as follows: d_{waist}: the diameter of the cylinder; l_{waist}: the length of the cylinder; r_{disk}: the radius of the disk; t_{disk}: the thickness of the disk.

We derive sufficient conditions to constrain each part of the object: the cylinder and the disk, respectively. The cylinder part cannot escape from the gap between both fingertips when its distance, $d_{\bar{L}+1}(\bar{\boldsymbol{\theta}}) \left(= \|\boldsymbol{p}_{2,\bar{L}+1}(\bar{\boldsymbol{\theta}}) - \boldsymbol{p}_{1,\bar{L}+1}(\bar{\boldsymbol{\theta}})\|\right)$, is less than the diameter of the cylinder:

$$d_{\bar{L}+1}(\bar{\boldsymbol{\theta}}) < d_{\text{waist}}. \tag{2}$$

The disk part cannot escape from the gap between both fingertips when $d_{\bar{L}+1}(\bar{\boldsymbol{\theta}})$ is less than the thickness of the disk:

$$d_{\bar{L}+1}(\bar{\boldsymbol{\theta}}) < t_{\text{disk}}. \tag{3}$$

In addition, the disk cannot escape between finger bodies when every distance between joints or the fingertips is less than the diameter of the disk:

$$d_{ijik}(\bar{\boldsymbol{\theta}}) < 2r_{\text{disk}} \quad (k \neq j) \tag{4}$$

$$d_{ij(i+1)k}(\bar{\boldsymbol{\theta}}) < 2r_{\text{disk}}, \tag{5}$$

$$(i = 1, \ldots, N) \quad (j = 1, \ldots, \bar{L}) \quad (k = 1, \ldots, \bar{L} + 1)$$

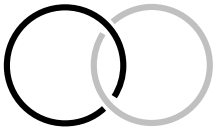


Fig. 6. Hopf link

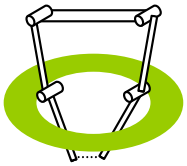


Fig. 7. Hopf link is not constructed

where d_{ijik} denotes the distance between the j-th joint of the i-th finger and the k-th joint of the i-th finger. The conditions to constrain the disk have been derived in [4].

Consequently, a series of cage-formed conditions for caging a dumbbell-like object is that (2), (3), (4) and (5) are satisfied.

E. In-cage Conditions

We derive *in-cage conditions*, which ensure that a target object is present in the region caged by a robot hand. Each series of in-cage conditions about three types of caging are derived.

1) Envelope-type Caging: An object is present in the region caged by a robot hand in cases of envelope-type caging when the center of the object is present inside a polyhedron whose vertices are joints and fingertips of the hand. Thus, extensively-used point-in-polyhedron tests can be utilize to examine whether the sufficient conditions are satisfied or not. Note that the polyhedron can be always formed because the robot hand is assumed to be a symmetrical hand (Sec. II-C).

2) Ring-type Caging: A ring-like object is present in the region caged by a robot hand in cases of ring-type caging when a hopf link (Fig. 6) is constructed by two closed curves: one goes through inside the object, and another is composed of a curve inside the hand and a line segment connecting both fingertips.

We assume that a closed curve inside an object is on a plane, and the closed region can be represented by $\mathcal{S}_{\text{obj}}$. When a hopf link is constructed, a robot hand or a line segment connecting both fingertips passes through $\mathcal{S}_{\text{obj}}$ odd times. If they pass through even times, a hopf link cannot be constructed such as Fig. 7.

Let us consider a case of caging a torus whose $\mathcal{S}_{\text{obj}}$ is a circle with r_{obj} radius. In the case, a closed curve of a robot hand goes through $\mathcal{S}_{\text{obj}}$ when the intersection of the closed curve and a plane including $\mathcal{S}_{\text{obj}}$ is within r_{obj} from the center of $\mathcal{S}_{\text{obj}}$.

3) Waist-type Caging: A dumbbell-like object is present in the region caged by a robot hand in cases of waist-type caging when a cylinder part of the object passes through a polygon whose vertices are joints and fingertips of the hand. The condition is satisfied when the center line of the cylinder goes through the polygon, and the distance between the intersection and the center of the cylinder is less than $l_{\text{waist}}/2$.

III. Motion Planning of Caging with a Robotic Arm/Hand

In this section, we show an algorithm for motion planning of 3D multifingered caging by a robotic arm/hand system. The algorithm is based on RRT (Rapidly exploring Random Trees) [5], which is a path planning method using random sampling. In the algorithm, it continue to connect randomly-sampled configuration points one after another until a goal configuration that satisfies the sufficient conditions for caging is found in many solutions.

A robotic arm (a manipulator) has L_{man} joints, and a vector of joint variables of the arm, $\boldsymbol{\Theta}$, can be expressed as follows:

$$\boldsymbol{\Theta} := [\Theta_1, \ldots, \Theta_{L_{\text{man}}}]^T, \in \Re^{L_{\text{man}}}, \tag{6}$$

where Θ_i denotes the joint variable of the i-th joint. With (1) and (6), we define a vector of robot configuration, z, as follows:

$$z := [\bar{\boldsymbol{\theta}}^T, \boldsymbol{\Theta}^T]^T \in \Re^{\bar{L}+L_{\text{man}}}. \tag{7}$$

The robot configurations in our caging problems have relatively high degrees of freedom (DOF). In addition, there are multiple goal configurations because of the sufficient conditions described as some inequalities with the joint variables. RRT, which we adopt to planning, is specially designed to handle nonholonomic constraints and high DOF.

A. Motion Planning Based on RRT

1) Procedure: Our planning procedure is almost based on the original RRT [5] and described as follows:

Step 1. Set an initial configuration: z_{ini} as a seed of a configuration path branches.
Step 2. Generate a random configuration, z_{rand}.
Step 3. Find the nearest configuration, z_{near} in the current configuration path branches.
Step 4. Generate a candidate of new configuration, z_{cand}, which is located between z_{rand} and z_{near}.
Step 5. Examine whether the robot collides with the target object and any obstacles at z_{cand}.
Step 6. When no collisions are detected, the candidate configuration becomes a new configuration, z_{new}.
Step 7. Generate a next candidate of new configuration between z_{rand} and z_{new} (Expand the branch).
Step 8. Repeat from Step 5 to Step 7 until any collisions are detected in Step 5, or until above repetition is done predetermined finite times, n_{rep}.
Step 9. Repeat from Step 2 to Step 8 until z_{new} satisfies both cage-formed conditions and in-cage conditions.

In Step 3, in order to determine the nearest configuration in the current path branches from a randomly-sampled configuration, z_{rand}, we calculate a norm between a configuration in the branches, z and z_{rand}, d_{cfg} defined with the following equation:

$$d_{\text{cfg}}(z, z_{\text{rand}}) := \sum_{j=1}^{\bar{L}} \left(\frac{(\bar{\theta}_j - \bar{\theta}_{\text{rand}})^2}{\dot{\theta}^2_{\text{max},j}} \right) + \sum_{j=1}^{L_{\text{man}}} \left(\frac{(\Theta_j - \Theta_{\text{rand}})^2}{\dot{\Theta}^2_{\text{max},j}} \right), \tag{8}$$

where, $\dot{\theta}_{\text{max},j}$ and $\dot{\Theta}_{\text{max},j}$ is maximum joint velocity of j-th joint of the finger and the manipulator, respectively. This normalization depends on the potential of each actuators and the norm, $d_{\text{cfg}}(z, z_{\text{rand}})$, means the square sum of the time to move from z to z_{rand} with each maximum joint velocity.

When a configuration in the current branches, z, gives the minimum d_{cfg}, the z becomes z_{near}.

In Step 4, a candidate configuration of the branches, z_{cand}, is arranged between z_{rand} and z_{near} with Linear Interpolation Method as follows:

$$z_{\mathrm{cand}} := z_{\mathrm{near}} + a_{\mathrm{br}} \frac{z_{\mathrm{rand}} - z_{\mathrm{near}}}{\|z_{\mathrm{rand}} - z_{\mathrm{near}}\|}, \tag{9}$$

where a_{br} is an arbitrary positive constant number that determines the length of expanded branch.

In Step 5, we use PQP [13] to examine whether the robot has collision with the object and any obstacles or not. Because of difficulty from computation cost, we just examine a collision test about z_{cand}.

If no collision are detected, the candidate configuration, z_{cand}, can be added as a new configuration in the path branches, z_{new}, in Step 6.

To expand the branch longer, we set a next candidate configuration of the path branches, z_{cand}, that is located between z_{rand} and z_{new}, and process the procedure from Step 5 to Step 7. The extension of a branch may continue until any collisions are detected, or the number of repetition times reaches a predetermined number, n_{rep}.

When we find a goal configuration that satisfies sufficient conditions of caging, we can obtain a path of the robot motion for caging.

2) Biased Sampling with solving Inverse Kinematics: A method of random sampling is useful for path planning in a complex environment and high-dimensional configuration space, particularly cases of difficulty finding a goal configuration. It requires, however, large computational cost to search wide area in configuration space, which includes obviously-distant points from a goal configuration. Thus, we adopt a method of biased sampling to our planner. The objective of biased sampling is to move a robot hand toward a target object more efficiently, and to reduce wasteful sampling of configuration.

Our biased sampling is based on solving inverse kinematics (IK) of manipulator. A standard IK problems are solved with a determined posture, but in caging, a particular posture is not set because of multiple solutions in sufficient conditions of caging. Then, we give a desired position of the tool center point of the manipulator that is enough close to a target object to capture it, and give a desired orientation by using an algorithm to generate uniformly-distributed random unit quaternions is presented in [14].

Once we obtain a set of joint variables of the arm solving IK, the joint variables are utilized instead of a randomly-sampled configuration, z_{rand}, except the wrist rolling joint of the arm, $\Theta_{\bar{L}_{\mathrm{man}}}$. Note that sampling about joint variables of the robot finger and $\Theta_{\bar{L}_{\mathrm{man}}}$ are always given by using random sampling.

IV. Object Recognition using AR Picture Markers

To plan a robot path for caging a target object, the specifications of the object: the shape, the size, the position and orientation, are required. We propose a method for object recognition using AR picture markers, and ARToolKitPlus [11], [12].

Fig. 8. AR picture markers on the objects

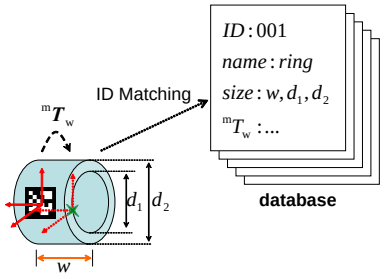


Fig. 9. Object properties

An AR picture marker, which is often used for Augmented Reality (AR), is described as a dotted pattern with square shape, and has 4096 patterns. We can embed a ID number in it to identify each object.

ARToolKitPlus is a successor to the popular ARToolKit [15], which is a software library for building AR applications including pose tracking libraries. ARToolKitPlus library has two valuable functions: recognizing AR markers and measuring the posture of the marker.

Our proposed procedure of object recognition with AR markers is as following steps.

Step 1. Recognize the AR marker attached to the target object (Fig. 8), and read the embedded ID number. At the same time, measure the posture of the marker.

Step 2. Load the the object properties from the database using the ID number (Fig. 9). They include the attachment location of the marker on the object, the shape and the size of the object and the objective pattern of caging. The posture of the object can be easily calculated.

Using AR markers has some advantages in object recognition. A robot does not have to measure the shape specification with a camera because they can be loaded from the database. Moreover, measuring the posture of the AR marker with ARToolKitPlus is easier than that of the object directly with a camera. These advantages make our proposed method useful in disordered scenes where objects tend to be hidden by any obstacles.

A. Object properties on the Database

Linking the ID numbers with each object listed on the database, we can easily recognize the object (Fig. 9). The database has the properties of each object: the attachment location of the marker on the object, the shape and the size of the object. Then the robot can recognize the details of the object, just reading the AR marker attached to the object

The object properties includes also patterns of objective caging for each object, Thus we can determine a particular pattern of caging for a target object from the database to plan a path of robot motion. Each pattern, which is classified in Sec. II-A, is with each series of sufficient conditions derived in Sec. II-B and [4].

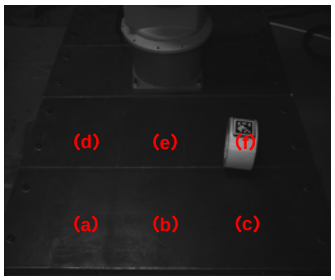


Fig. 10. Position measurement using ARToolKitPlus

TABLE I

ESTIMATED ERRORS IN POSITION MEASUREMENT

		(a)	(b)	(c)	(d)	(e)	(f)
Marker	x	400	400	400	200	200	200
location	y	−200	0	200	−200	0	200
[mm]	z	50	50	50	50	50	50
Estimated	x	4	1	5	5	5	5
errors	y	2	3	2	1	1	5
[mm]	z	1	2	1	1	1	1

B. Posture Measurement

We can measure the relative posture between a camera and a marker attached on the target object with a particular function of ARToolKitPlus [11]. After that, the relative posture between the object and the robot can be calculated with homogeneous transformation matrices immediately. It is because the location of the marker on the object can be known on the database.

We tested the accuracy of our proposed position measurement with AR markers and ARToolKitPlus. The markers located in several points in the robot coordinate are 30×30 mm, and face to the camera almost oppositely (Fig. 10, Table I). The camera is FL2G-50S5M (Flea2), which has 2448×2048 pixels with a lens: JHF8M-5MP. In our experiments, estimated errors in position measurement using ARToolKitPlus are within 5 mm (Table I).

V. CAGING EXPERIMENTS

In this section, we show some experimental results of our caging procedure among object recognition, planning and execution. The intended patterns of caging in this paper are only ring-type and waist-type, which can be achieved by a two-fingered hand. It is because some collisions between the fingers and obstacles such as a floor often occur in cases of envelope-type caging, and our proposed method cannot deal with such problems.

A. Experimental Settings

Our robotic arm/hand system to perform caging experiments consists of a manipulator that has six DOF and a robot hand that comprises two fingers with two joints each. The joints of both the arm and the hand are position-controlled. In addition, there are no force sensors and/or tactile sensors to execute caging constraint. The camera as same as in Sec. IV-B is utilized to recognize the target object using the AR picture marker attached to the object.

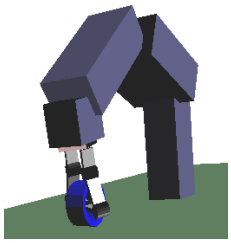


Fig. 11. A planning result of caging the cylinder

Fig. 12. Caging the cylinder by the arm/hand system

There are four target objects: a cylinder (with a hollow), a ring, a mug-like object (every above case is ring-type caging) and a dumbbell (waist-type caging). The cylinder is with inner radius: 38 mm, outer radius: 50 mm and height: 50 mm. The ring is a torus with the radius from the center of the hole to the center of the ring: 124 mm; the diameter of the ring: 16 mm. The mug-like object (a cylinder solid with a handle) is with the radius from the center of the circular handle to the center of the handle body: 124 mm; the radius of the handle: 8 mm; the diameter of the cylinder: 100 mm and the height of the cylinder: 150 mm. The dumbbell is with $r_{\text{disk}} = 45$ [mm], $t_{\text{disk}} = 1$ [mm], $d_{\text{waist}} = 26$ [mm] and $r_{\text{waist}} = 222$ [mm]. We test two different situations for each object, where the position and orientation of the object are changed.

Motion planning is performed on a Linux PC whose CPU is Intel Core i7 running at 2.8 GHz.

B. Planning and Execution Results

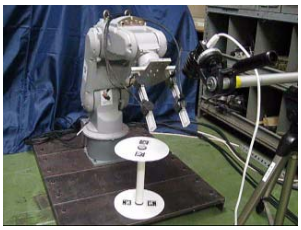
In each scene, the robot can recognize the AR picture marker attached to the object and access to the database to load the object properties. Additionally, the posture of the marker can be measured using the libraries of ARToolKitPlus, and then, that of the object can be easily calculated. The planner can produce each path of robot motion for caging the object with each series of sufficient conditions (Fig. 11). The average computation time for each planning in ten trials are shown in Table II.

The planned motions for caging are successfully executed by the robotic arm/hand system (Fig. 12, 13, 14). In the planned motions, the robot does not collide with the object, however, in the experiments, the robot sometimes touched the object because of some errors of position estimation (Sec. IV-B). In some scenes, caging the object could be accomplished even when the robot touched the object before caging. In other cases, the execution failed because the object touched by the robot moved from the correct position.

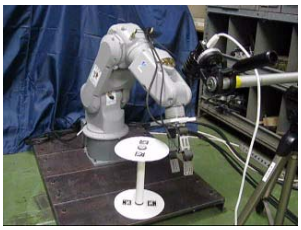
To avoid the practical collision, we test collision checks with a certain amount of margins, that is, we use a little larger model of the object to plan. The planner can also produce each robot motion for caging, in which the robot hand locates farther from the object than in previous cases without margins. Furthermore, the actual robot can execute the new planned caging motion without collision with the object (Fig. 15). The average computation time for each

Fig. 13. Experiment: caging the ring

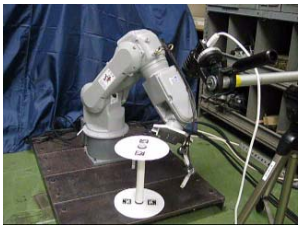
Fig. 14. Experiment: Caging the mug-like object



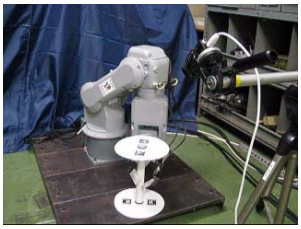
(a)



(b)



(c)



(d)

Fig. 15. Planned motion for caging the dumbbell (with object margins)

TABLE II

AVERAGE PLANNING TIME [S] (TEN TRIALS EACH)

		Cylinder	Ring	Mug	Dumbbell
(w/o margins)	Position 1	10	4	12	160
	Position 2	41	3	45	21
(w/ margins)	Position 1	330	32	77	456
	Position 2	144	4	82	245

planning in ten trials are also shown in Table II. Every computation time in the case with considering margins is larger than those without it. It is because that enlarging the model of the robot hand causes decreasing the space where the hand can reach, and required path becomes less found.

C. Discussion

Our current planner has not been applied to envelope-type caging (Sec. II-E.1) because it cannot deal with the cases that the hand may contact with the object or obstacles. It is difficult to execute envelope-type caging without any collisions, for example, caging a sphere on the floor. If we put the sphere on the pedestal, we can applied our planner to the caging, although it is a little arbitrary. In the case of caging the ring (Fig. 13), we use a pedestal to facilitate approaching to the hollow part of the ring.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we improved a motion planner for 3D multifingered caging with the actual robotic arm/hand system. The parts of sufficient conditions for three caging patterns including four objects were derived. An object recognition using AR picture marker is presented to provide the set of properties of the object, which is linked to the marker and loaded from a database. The posture of the object can be calculated with measuring that of the marker. Planning of robot motions with the object recognition for four objects and execution of the planned motions with the arm/hand system were successfully performed. We also examined the advantages of the margins considered in planning to avoid collisions in practical execution caused by the errors of posture estimation.

In future works, large planning time should be reduced. Additionally, more various objects and scenes should be investigated to adopt 3D multifingered caging to daily tasks.

VII. ACKNOWLEDGMENTS

This work was partly supported by Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B), No. 22700200.

REFERENCES

[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, U.S.A., April 2000, pp. 348–353.

[2] 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, March 1999.

[3] R. Diankov, S. S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning with caging grasps," in *Proc. of IEEE/RAS Int. Conf. on Humanoid Robots*, Daejeon, Korea, December 2008, pp. 285–292.

[4] S. Makita and Y. Maeda, "3d multifingered caging: Basic formulation and planning," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and System*, Nice, France, 2008, pp. 2697–2702.

[5] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. TR98-11, 1998.

[6] P. Pipattanasomporn and A. Sudsang, "Two-finger caging of concave polygon," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Orlando, FL, U.S.A., May 2006, pp. 2137–2142.

[7] 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, December 2007.

[8] 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, November/December 2008.

[9] P. Pipattanasomporn, P. Vongmasa, and A. Sudsang, "Caging rigid polytopes via finger dispersion control," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, U.S.A., May 2008, pp. 1181–1186.

[10] Z. Wang and V. Kumar, "Object closure and manipulation by multiple cooperating mobile robots," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Washington D.C., U.S.A., May 2002, pp. 394–399.

[11] D. Schmalstieg, "ARToolKitPlus," http://handheldar.icg.tugraz.at/artoolkitplus.php.

[12] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," in *Computer Vision Winter Workshop*, 2007.

[13] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Computer Science Dept., University of North Carolina, Chapel Hill, Tech. Rep. TR99-018, 1999.

[14] J. J. Kuffner, "Effective sampling and distance metrics for 3d rigid body path planning," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA, U.S.A., April 2004, pp. 3993–3998.

[15] P. R. Lamb, "ARToolKit," http://www.hitl.washington.edu/artoolkit/download/.