Motion Planning of Multiple Mobile Robots for Cooperative Manipulation and Transportation

Atsushi Yamashita, *Member, IEEE*, Tamio Arai, *Member, IEEE*, Jun Ota, *Member, IEEE*, and Hajime Asama, *Member, IEEE*

***Abstract*—In this paper, we propose a motion-planning method of multiple mobile robots for cooperative transportation of a large object in a three-dimensional environment. This task has various kinds of problems, such as obstacle avoidance and stable manipulation. All of these problems cannot be solved at once, since it would result in a dramatic increase of the computational time. Accordingly, we divided the motion planner into a global path planner and a local manipulation planner, designed them, and integrated them. The aim was to integrate a gross motion planner and a fine motion planner. Concerning the global path planner, we reduced the dimensions of the configuration space (C-space) using the feature of transportation by mobile robots. We used the potential field to find the solution by searching in this smaller-dimension reconstructed C-space. In the global path planner, the constraints of the object manipulation are considered as the cost function and the heuristic function in the A^* search. For the local manipulation planner, we developed a manipulation technique, which is suitable for mobile robots by position control. We computed the conditions in which the object becomes unstable during manipulation and generated each robot's motion, considering the robots' motion errors and indefinite factors from the planning stage. We verified the effectiveness of our proposed motion planning method through simulations.**

***Index Terms*—Cooperative manipulation, cooperative transportation, motion planning, multiple mobile robots.**

I. Introduction

Mobile robots are expected to undertake various tasks in manufacturing plants, warehouses, and construction sites. In order to improve task flexibility and fault tolerance, the concept of cooperation by multiple mobile robots was proposed [1]. In the future, mobile robots should work in a real three-dimensional (3-D) environment. In this complicated situation, a good motion planning method is very important to accomplish tasks efficiently. In particular, there is significant demand for robots that can carry out a transport task. Therefore, we propose a motion planning method for the cooperative transportation of a large object by small multiple mobile robots in a 3-D environment. This task, however, has various kinds of problems. For example, we must plan paths of the robots and the object to avoid obstacles, construct a stable manipulation method, and decide the motions that the robot will undertake (Fig. 1). The robots must also complete the transportation task quickly. All of these problems cannot be solved at once, since it would result in a significant increase in the computational time. Conventional path planning methods consider only geometrical and topological conditions, such as the shapes of obstacles and robots. They do not take into account the statics of the object when robots manipulate it. Accordingly, we divided the motion planner into a global path planner and a local manipulation planner. The former plans the paths of the object and robots and considers the geometrical conditions. The latter determines how to manipulate the object and considers the statics. In other words, we aimed at integrating a gross motion planner and a fine motion planner.

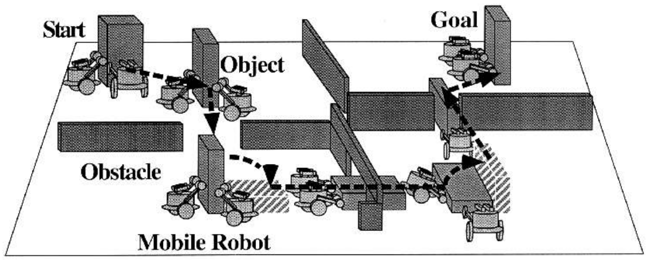


Fig. 1. Cooperative transportation by multiple mobile robots in a 3-D environment. Robots must avoid obstacles, manipulate the object stably, and reach the goal quickly.

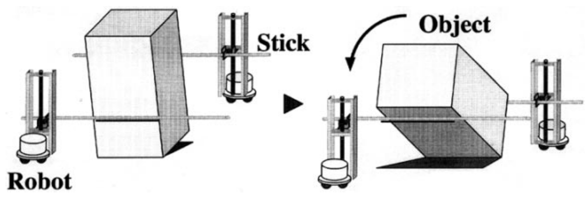


Fig. 2. Tumbling manipulation by multiple mobile robots.

The generic problem of manipulation is known to be very complex and difficult to solve. This paper proposes a method to solve a specific problem of manipulation. Concerning the manner in which manipulation is to take place, we propose that robots manipulate an object by pushing with sticks in a multiple mobile robot system (Fig. 2). The robots tumble the object cooperatively to change its posture. This specificity makes the problem easier to solve.

Concerning path planning, we aimed at obtaining a solution that would combine low computation costs and feasibility. Feasibility means that planned robot motions are in agreement with

Manuscript received March 31, 2001; revised March 27, 2002 and September 5, 2002. This paper was recommended for publication by Associate Editor L. Kavraki and Editor S. Hutchinson upon evaluation of the reviewers' comments. This paper was presented in part at the IEEE International Conference on Robotics and Automation, San Francisco, CA, April 2000.

A. Yamashita is with the Department of Mechanical Engineering, Shizuoka University, Shizuoka 432–8561, Japan (e-mail: yamashita@ieee.org).

T. Arai and J. Ota are with the Department of Precision Engineering, The University of Tokyo, Tokyo 113–8656, Japan (e-mail: arai@prince.pe.u-tokyo.ac.jp; ota@prince.pe.u-tokyo.ac.jp).

H. Asama is with RACE (Research into Artifacts, Center for Engineering), The University of Tokyo, Tokyo 153-8904, Japan (e-mail: asama@race.u-tokyo.ac.jp).

Digital Object Identifier 10.1109/TRA.2003.809592

their real surroundings (for example, robots never execute a dangerous manipulation imprudently). We also make the assumption that the robots can move in any direction (for example [2]) and always keep in contact with the object when they transport it. Moreover, they can transport a single polyhedral object. In our approach, the dimension of freedom (DOF) of the configuration space (C-space) is reduced using the feature of transportation tasks because the original DOF of C-space becomes too large (The DOF of the object is six and the DOF of the robots is $3r$ when r is the number of robots). Therefore, we reduce the DOF of the C-space by using decoupled motions and restricting the robots to a given formation relative to the object. The $(6+3r)$ DOF is reduced to 5DOF (see Section IV).

Our method plans the collision-free motions of the robots and the object. In other words, the positions and end-effectors' motions of the robots and the position and orientation of the object from the start state to the goal state are continuously planned when the geometry of all things and the property of the object (mass, center of mass, and coefficient of friction) are given.

The composition of this paper is detailed below. The next section reviews previous research, and Section III describes the outline of our motion planning method. In Section IV and V, the global path planner and the local manipulation planner are explained. In Section VI, we verify our method with simulations. The conclusions are discussed in Section VII.

II. Previous Studies

Many studies have focused on motion planning for the classical movers' problem in a 3-D environment. This problem is very difficult because of the high-dimensional C-space [3], [4]. A brute force search can solve problems with a low-dimensional C-space (for example, path planning of one mover in a two-dimensional (2-D) environment). However, a simple planner cannot solve high-dimensional problems (for example, in a 3-D environment, the DOF of the C-space is at least six). There have been many studies about a path planner with a high-dimensional C-space [5], [6]. Kondo used multi-heuristics by searching [7]. Barraquand *et al.* constructed a randomized path planner [8], [9], and Gupta *et al.* proposed a backtracking method [10] by solving problems with many degrees of freedom. Hwang *et al.* proposed a resolution-complete and efficient method [11], and Kavraki *et al.* proposed a probabilistic roadmap method [12]–[14]. Terasaki *et al.* proposed the motion planning method of a two-fingered gripper [15]. These methods can efficiently find the path of an object or a robot, and their purpose is to find a path in the C-space to avoid obstacles. In transportation tasks, however, we need to consider the motions of the workers or robots that transport and manipulate an object, but these planners do not take into account the workers' or robots' motions. Therefore, the generated paths (motions) may be difficult to realize, for there is no consideration about the stability of an object and how to transport and manipulate it. Concerning the cooperative manipulation of polyhedral objects, Erdmann proposed two-palm nonprehensile manipulation [16]. Lynch proposed toppling manipulation that used a much simpler robot motion to knock a object over a new face [17]. However, their analyzes does not explicitly compute the actual contact forces, and the considered workspace in this research is smaller than in cooperative transportation tasks. When the constraints of an object's motion must be considered, the ordinary C-space is so large that the computational costs increase dramatically. Therefore, in our approach, we reduced the DOF of the C-space.

There have also been many studies on cooperative transportation and the manipulation of objects with a multiple mobile robot system. Rus *et al.* built a method to push and transport an object using sensor information [18]. Their method is based on the knowledge obtained conventionally in the research field of pushing manipulation [19]–[23]. Hashimoto *et al.* proposed a control scheme to transport a palletized load using mobile robots [24]. In their research, the object is connected to the robots by joints; then, object manipulation cannot be executed. Sugar *et al.* proposed a control algorithm for cooperative transportation by a multiple mobile manipulator without a special purpose fixture [25]. Kosuge *et al.* aimed at transporting an object by lifting and adopted the feedback control method using the information of robots' force sensors [26]. Khatib *et al.* controlled the inner force that is applied to an object and adopted a stable handling strategy to compensate the motion errors of robots [27]. Sawasaki *et al.* realized the tumbling manipulation technique of an object with two mobile manipulators [28]. As valuable as these studies are, they only aim at constructing a control method and do not consider the avoidance of obstacles while transporting the object. Ota *et al.* discuss the motion planning of mobile robots that transfer a large object cooperatively while avoiding collision [29], yet in a 2-D environment. Concerning transportation tasks in a 3-D environment, Osumi coped with the unevenness of the ground [30], and Hara *et al.* proposed a control method for the task by legged robots [31], [32]. However, these studies could not change the object's posture actively to avoid obstacles. In brief, a motion planning method by multiple mobile robots for cooperative transportation and manipulation has not been proposed until now.

III. Transportation Task by Mobile Robots

A. *Outline of the Proposed Motion Planner*

The cooperative transportation task includes the problems of avoiding collisions and achieving stable manipulation. The robots must manipulate an object and change its posture when the path is so narrow that the robot cannot pass through it.

In our method, we followed the local experts [33] to plan the manipulation method and divided a motion planner for a transportation task into a global path planner phase and a local manipulation planner phase (Fig. 3). The global planner decides when and where to manipulate the object and what kind of manipulation way can be realized based on the result of the local manipulation planner. It outputs the paths of the objects and the robots with obstacle avoidance. The local planner outputs a method for and information about the manipulation. The manipulation way indicates the motions of an object and robots to change the posture of the object safely. The information about the manipulation indicates the manipulation space, i.e., the area necessary to manipulate the object and the manipulation cost,

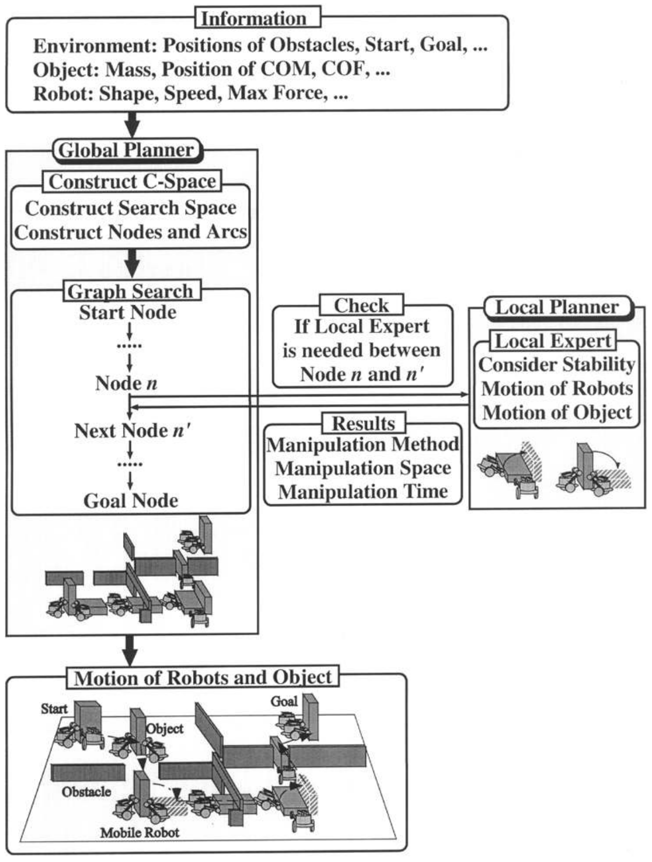


Fig. 3. Outline of our proposed motion planner.

i.e., the necessary time to finish the manipulation. The information about the manipulation is utilized to plan the global motions of the objects and the robots.

B. The Global Path Planner

The biggest problem concerning the global path planner is the dramatic increase in the computation time that results from the high-dimensional C-space. In this paper, we reconstructed the C-space and reduced the dimension of the C-space by considering the features of the transportation task by mobile robots. Furthermore, we used the potential field defined in the real world space to find a solution by searching in this low-dimensional C-space. Constraints in manipulating were considered as the work area and the potential function. The work area was computed as the manipulation space, and the potential function was computed as the manipulation cost in the local planner.

C. The Local Manipulation Planner

Concerning the local manipulation planner, one of the biggest problems when mobile robots work is the effect of the robots' position errors. In previous studies, the force sensor's information and the image information from the camera were fed back to correct the motion errors. The motion errors of mobile robots were avoided with the on-line method. In addition, excessive inner force applied to an object was avoided by a force control approach. In other words, a feedback control approach was adopted to manipulate an object by a multiple mobile robot system. However, it is difficult for mobile robots to measure force correctly and to manipulate an object with a force-control method like fixed manipulators while they move. Then, in this research, we propose a robust manipulation planning method for the motion errors by taking into consideration the motion errors in the motion planning stage. We aim at constructing a planning method of manipulation that is suitable for mobile robots under position control [34], [35].

D. Manipulation and Transportation Style

We propose a multiple mobile robot system in which robots manipulate an object by one of several manipulation methods: carrying the object, pushing the object to reorient it, tumbling the object to change the face by pushing with sticks (Fig. 2), and so on. When the manipulating tasks are carried out, the stability of operation is improved because the contact area to the object becomes larger [36], [37]. The robots tumble the object so that the face that contacts the floor changes. By repeating this tumbling operation, it is possible to change the posture of an object arbitrarily. Our idea is related to Lynch's toppling manipulation [17] that the robot manipulate objects by simple motions. To integrate two planners, the constraints of the object manipulation are considered in the path planner.

Position errors occur when position-controlled mobile robots move. Therefore, there is a risk that multiple mobile robots apply excessive inner force to an object when they touch the object at the same time. Then, in this paper, more than two sticks cannot touch the object at any time to avoid excessive inner force when the robots move. Since it is difficult for mobile robots to lift up a large or heavy object while they change its posture, they manipulate it without resorting to the lifting-up operation. Therefore, the object always contacts the environment (the floor) during a toppling motion. The robots can move the sticks up and down, using pushing and tumbling operations.

The robots transport the object by using a carrying motion. This is because they cannot carry it in a long distance stably by using a pushing motion that is influenced strongly by the floor's state. While transporting the object, robots do not change their positions with respect to the object. In other words, they do not change their original formation. The robots depart from the object when they manipulate it and stay close to the object when they transport it.

There are four primitive operations in total. The robots change the object's position in the world coordinate with a position change operation, Fig. 4(a). The robots change the object's orientation with an orientation change operation, Fig. 4(b). In these two operations, the robots cannot change their positions in the object coordinate. The robots change their position where they handle the object with an arrangement change operation, Fig. 4(c). In this operation, the object's position or orientation is never changed. The robots manipulate the object and change its posture with a face change operation, Fig. 4(d).

The robots accomplish the transportation task with four primitive operations. The local manipulation planner deals with the face change operation, and the global path planner deals with the other operations.

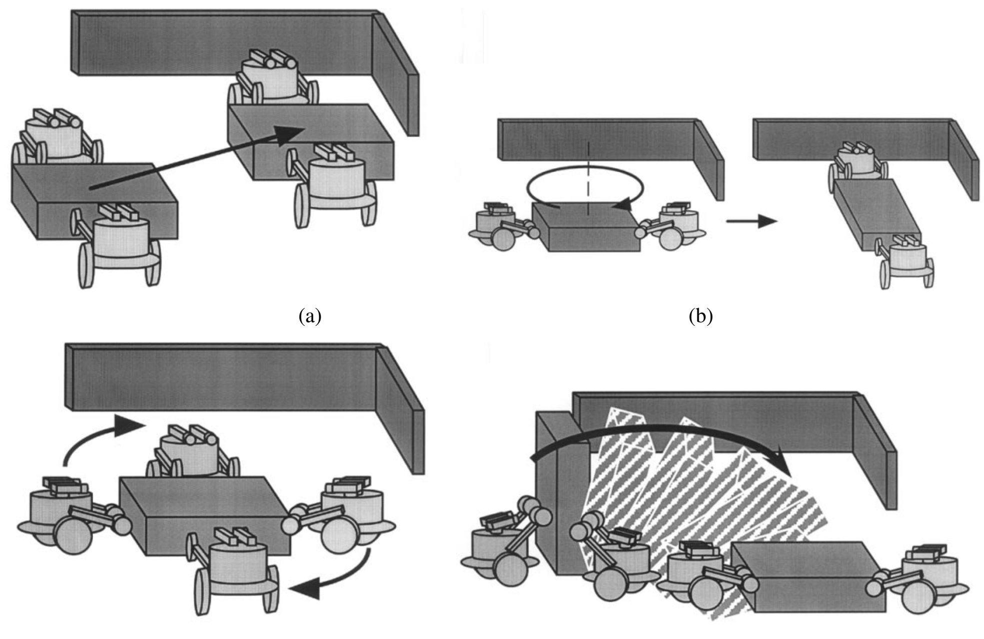


Fig. 4. Four primitive operations of an object by multiple mobile robots. (a) Position change operation. (b) Orientation change operation. (c) Arrangement change operation. (d) Face change operation.

E. Problem Settlement

We make the following assumptions for motion planning.

1) Environment
   a) 3-D environment with obstacles.
   b) The obstacles are represented as polyhedrons.
   c) The shape, position, and orientation of each obstacle are known.
2) Object
   a) The object is represented as an extruded polyhedron.
   b) The geometry of an object and the location of its center of mass are known.
   c) The coefficients of friction between the object and the floor and between the object and the stick are known.
3) Mobile Robot
   a) The robots can move in all directions.
   b) The robots transport the object around it by using a carrying motion.
   c) Robots have lift-up mechanisms to control their end-effectors' height (contact points' height).
4) Manipulation
   a) The robots tumble the object while one edge (this edge is the center of rotation) always contacts the floor.
   b) All motions are quasi-static.
   c) All frictional interactions of the object are described by Coulomb's law of friction.

F. Definition of Problem

In this paper, we propose a motion planning method of multiple mobile robots for cooperative transportation in a complicated 3-D environment. The definition of the problem is as follows;

We determine the robot motions with avoidance of obstacles when the initial and goal configuration of the robots and the object are given on the condition that all geometric information about the environment is known. To transport and manipulate the object safely, a robot motion that realizes the object trajectory without colliding with obstacles or losing stability must be calculated. However, motion errors by robots do occur, and robots cannot move according to precise orders.

We define the problem according to the description of C-space in [3].

When the object is described as A_{object}, the robots are described as $A_{\text{robot}} = (A_{\text{robot},1}, A_{\text{robot},2}, \ldots, A_{\text{robot},r})$ (r: number of robots), and $A = (A_{\text{object}}, A_{\text{robot}})$ is expressed as the movable object in Euclid space W. The workspace is given as R^3. $B_1, B_2, \ldots, B_p$ are the obstacles in the environment (p: number of obstacles). The shapes of $A, B_1, \ldots, B_p$ and the positions and orientations of B_i in W are known.

We define C_R and C_O to represent the robots' C-space and the object's C-space, respectively. The dimension of C_R is $3r$ and that of C_O is 6.

C-Obstacle $CB_{R,i}$ and $CB_{O,i}$ (a set of the obstacles in C-space) are given in (1)–(2), and the free space $C_{R,\text{free}}$ and $C_{O,\text{free}}$ are given in (3)–(4).

$$CB_{R,i} = \{q \in C_R \setminus A(q) \cap B_i \neq 0\} \tag{1}$$

$$CB_{O,i} = \{q \in C_O \setminus A(q) \cap B_i \neq 0\} \tag{2}$$

$$C_{R,\text{free}} = C_R \setminus \bigcup_{i=1}^{p} CB_{R,i} \tag{3}$$

$$C_{O,\text{free}} = C_O \setminus \bigcup_{i=1}^{p} CB_{O,i}. \tag{4}$$

Motion Planning Problem: The motion planning problem is to gain a continuous function that satisfies (5) under Conditions 1–3 ((6)–(8)) when the initial configuration $q_{\rm init} = (q_{\rm object,init}, q_{\rm robot,init})$ and the goal configuration $q_{\rm goal} = (q_{\rm object,goal}, q_{\rm robot,goal})$ are given. In these equations, $q_{\rm object}(t)$ is the object's configuration in time t, and $q_{\rm robot}(t) = (q_{\rm robot,1}(t), q_{\rm robot,2}(t), \ldots, q_{{\rm robot},r}(t))$ is the robots' configuration in time t.

$$t \in [0,T] \mapsto q_{\rm robot}(t) \in C_{R,\rm free} \tag{5}$$

where

$q_{\rm robot}(0) = q_{\rm robot,init}$;

$q_{\rm robot}(T) = q_{\rm robot,goal}$;

T time to reach the goal configuration ($T > 0$).

Condition 1: We express that $q^*_{\rm robot}(t) = (q^*_{\rm robot,1}(t), q^*_{\rm robot,2}(t), \ldots, q^*_{{\rm robot},r}(t))$ is the robot configuration when the robots moves in real, and $q_{\rm error} = (q_{\rm error,1}, q_{\rm error,2}, \ldots, q_{{\rm error},r})$ is the accuracy (motion error) of the positioning of robots. $q^*_{\rm robot}(t)$ is the real trajectories (motions) of the robots while $q_{\rm robot}(t)$ is their planned trajectories. Equation (6) must be satisfied while $0 < t < T$.

$$|q^*_{{\rm robot},i}(t) - q_{{\rm robot},i}(t)| \leq q_{{\rm error},i}. \tag{6}$$

Condition 2: The continuous function $q_{\rm object}(t)$ (object's motion), which is realized by the continuous function $q_{\rm robot}(t)$ (robots' motion) that satisfies (5), must satisfy (7).

$$t \in [0,T] \mapsto q_{\rm object}(t) \in C_{O,\rm free} \tag{7}$$

where

$q_{\rm object}(0) = q_{\rm object,init}$;

$q_{\rm object}(T) = q_{\rm object,goal}$.

Condition 3: The object's motion (the methods of object manipulation and transportation) that is generated by the robots' motion is expressed as OP_j. OP_j indicates manipulation method that changes the object configuration from $q_{\rm object}(t_j)$ to $q_{\rm object}(t_{j+1})$ and is defined in (8). OP_j means four primitive operations shown in Fig. 4 and the object configuration can never change without OP_j. $q^*_{\rm object}(t)$ is the real trajectory (motion) of the object while $q_{\rm object}(t)$ is its planned trajectory.

$$OP_j : t \in [t_j, t_{j+1}] \mapsto q_{\rm object}(t) \in C_{O,\rm stable} \tag{8}$$

and we require

$$\begin{aligned} q_{\rm object}(t_j) &\in C_{O,\rm stable} \\ q^*_{\rm object}(t_j) &\in C_{O,\rm stable} \\ q_{\rm object}(t_{j+1}) &\in C_{O,\rm stable} \\ q^*_{\rm object}(t_{j+1}) &\in C_{O,\rm stable} \end{aligned}$$

where

$C_{O,\rm stable}$	set of stable object configurations (the subset of C_O);
$OP_j(t_j) = q_{\rm object}(t_j)$	configuration when a new primitive operation starts;
$OP_j(t_{j+1}) = q_{\rm object}(t_{j+1})$	configuration when a primitive operation ends.

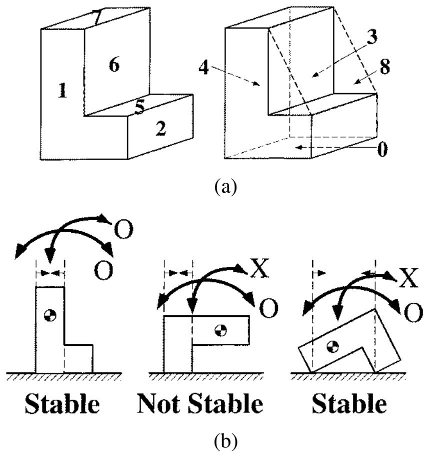


Fig. 5. Object representation. (a) Object face. (b) Possible manipulation.

In other words, the motion planning problem is to calculate the order of OP_j ($j = 0, \ldots, s-1$, $t_0 = 0$, $t_s = T$) and generate the robot configuration to realize each OP_j.

IV. Global Motion Planning

A. Outline of Global Planner

In the global path planner, the paths of the objects and robots with avoidance of obstacles can be found. The global planner decides when and where to manipulate the object and what kind of manipulation way can be realized based on the result of the local manipulation planner. At first, the planner computes the possible manipulation (Section IV-B). In the next step, we reduce the DOF of C-space and discretize each reduced DOF. Concerning the DOF of the environment, we use an octree method, and, concerning the other DOF, we discretize it in a certain resolution (Section IV-C). After obtaining the discrete C-space, we adopt a A^* search to find a path because it can obtain an optimal solution in a certain degree.

B. Possible Manipulation

Before constructing the C-space and the graph search in the global planner, we consider all possible manipulations. At first, we compute the state in which an object can be grounded. A face number represents the grounded state. For example, in the case of the object shown in Fig. 5(a), when Faces 0–4, and 7 (ordinary faces) and Face 8 (virtual face) touch the floor, the object can be grounded (In this state, robots can transport an object). If Faces 5 and 6 cannot touch the floor, the object cannot be grounded. Next, we check if the object can safely be manipulated from one state to another by tumbling.

We use matrix $M = (m_{ij})$ to express whether manipulation is possible or not. If the manipulation from stable Face i to stable Face j is possible, $m_{ij} = 1$. If it is impossible, $m_{ij} = 0$. However, if Face j is an unstable face because of the position of the center of gravity, the manipulation between stable Face i and unstable Face j is possible as the intermediary manipulation, and $m_{ij} = 1$ (Fig. 5(b) center).

C. Reconstruction of Configuration Space

1) Reduction of DOF: We deal with the problem that multiple mobile robots transport an object around it. The DOF of

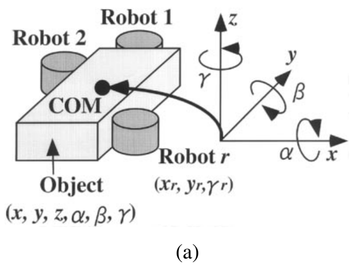
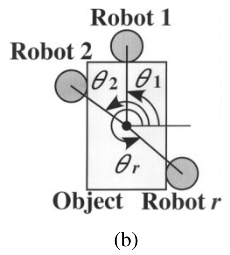



Fig. 6. Model of the object and the robots. (a) DOF of the object and the robots. (b) Arrangement of the robots in the object coordinate.

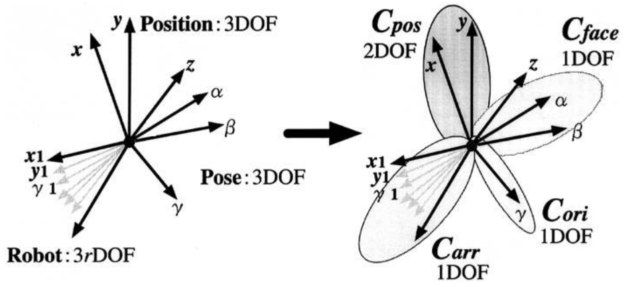


Fig. 7. Reduction of DOF.

the object is six $(x, y, z, \alpha, \beta, \gamma)$, and the DOF of robots is $3r$ (x_m, y_m, γ_m) $(m = 1, \ldots, r$, r: the number of robots). The DOF of the C-space is $(6 + 3r)$, as shown in Fig. 6(a).

Because of the high-dimensional C-space, we reduce the dimensions of the C-space using the feature of transportation with mobile robots. In a transportation task with mobile robots, the possible paths and motions are limited and are different from the paths in the problems of free flying objects. Robots cannot transport the object while changing its posture in a complicated way. Therefore, they change the object's posture from one grounded state in which one face of the object contacts the floor to another grounded state. These states are abstracted as characteristic configurations, and the planner uses these configurations to find a practical path.

The positions and orientations of the robots can be considered as an arrangement of the robots in the object coordinate, Fig. 6(b).

The reconstructed C-space is shown in Fig. 7. In Fig. 7, $C_{\rm pos}$ corresponds (x, y) in Fig. 6(a) (2DOF). The variable (v_x, v_y) about $C_{\rm pos}$ axis can be changed by the position change operation, Fig. 4(a). When v_x changes, the position of the object's reference point about the x axis changes, and, when v_y changes, the position about the y axis changes. The relative relationship between the robots and the object does not change.

$C_{\rm ori}$ corresponds γ in Fig. 6(a) and expresses it as the orientation of the object and the robots (1DOF). The variable v_γ about the $C_{\rm ori}$ axis can be changed by the orientation change operation, Fig. 4(b). When v_γ changes, the orientation of the object's reference point about the z axis changes. The relative relationship between the robots and the object does not change.

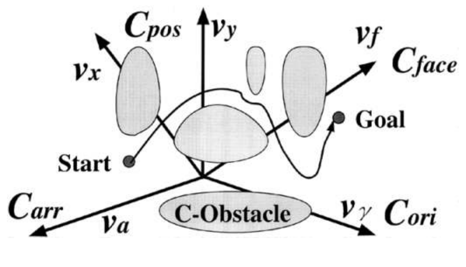


Fig. 8. Path planning in the reconstructed C-space.

$C_{\rm arr}$ corresponds to $(\theta_1, \theta_2, \ldots, \theta_r)$ in Fig. 6(b) and expresses them as the arrangement (formation) of mobile robots (1DOF). The variable v_a about the $C_{\rm arr}$ axis can be changed by the arrangement change operation, Fig. 4(c). When v_a changes, the formation of the robots (the relative relationship between the robots and the object) changes. The position and orientation of the object do not change.

$C_{\rm face}$ corresponds (α, β) and z in Fig. 6(a) and expresses them as the face of the object contacting the floor (1DOF). The variable v_f about the $C_{\rm face}$ axis can be changed by the face change operation, Fig. 4(d). When v_f changes, the object's face that touches the ground changes.

Accordingly, $(6 + 3r)$DOF can be reduced to 5DOF. The whole reduced C-space $C_{\rm all}$ is expressed as follows: $C_{\rm all} = (C_{\rm pos}, C_{\rm ori}, C_{\rm arr}, C_{\rm face})$ (Fig. 8).The collision-free path from the start configuration to the goal configuration can be searched in this continuous C-space. However, the computation costs to search for the path are larger in the continuous C-space than in the discrete C-space. Therefore, after reducing the DOF of C-space, we discretize each DOF. Concerning the $C_{\rm pos}$, an octree method is adopted. The other DOF (IV-C.2) is discretized in a certain resolution (IV-C.3). Eventually, a graph that consists of nodes and arcs is generated (IV-C.4).

2) Representation of the Environment: In our method, all things (object, robots, and obstacles) are represented by an octree, which is the approximate cell decomposition method for a 3-D environment. By using the octree method to represent them, all things can be dealt with in the same way, and the number of cells is small compared with that in the exact cell decomposition method. When we solve problems with a high-dimensional C-space, this representation is efficient and practical. In our method, we aim at a resolution-complete and efficient planner.

A 2-D environment represented by a quadtree is shown in Fig. 9(a), and a 3-D environment represented by an octree is shown in Fig. 9(b). Cells are divided into white ones (free space) and black ones (obstacles). The object and robots can be represented as one object because the formation of the robots is not changed while they are transporting the object, Fig. 9(c). As to the object and robots, cells where the object and robots exist are called the object cell $A_{{\rm obj},j}$ (j: cell number).

For representing the environment by an octree, $C_{\rm pos}$ is discretized. The reference point of the object (v_x, v_y) can only move from the present position (present node) to the adjacent cells in an octree representation. A path free from obstacles can

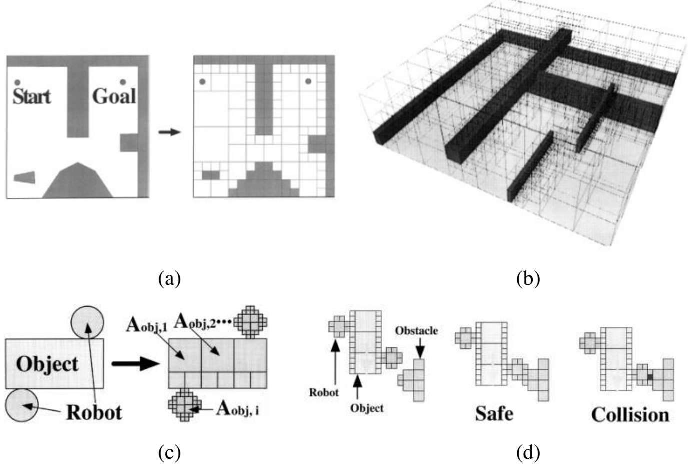


Fig. 9. Representation by approximate cell decomposition. (a) 2-D environment (quadtree). (b) 3-D environment. (c) Robots and the object. (d) Collision check between two cells.

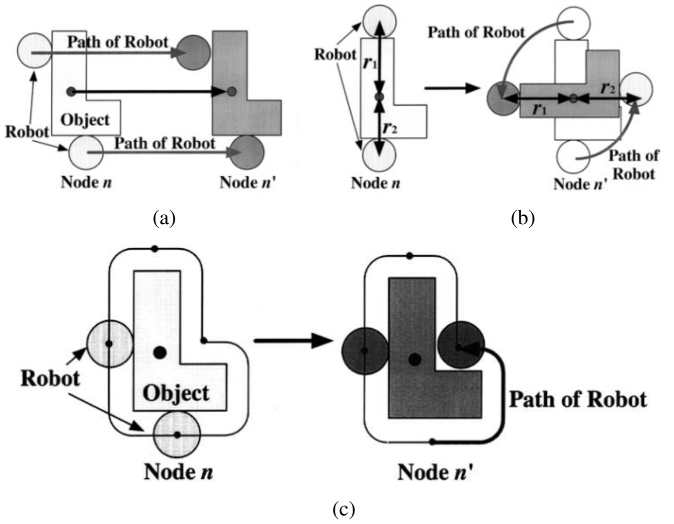


Fig. 10. Constraints and individual robot motions in the primitive operations. (a) Position change operation (C_{pos}). (b) Orientation change operation (C_{ori}). (c) Arrangement change operation (C_{arr}).

be found efficiently as a sequence of white cells from a start configuration to a goal.

3) Discrete Configuration Space: A discrete representation of C_{pos} is obtained by an octree. We discretize the other DOF and explain the constraints and the individual robot motions in each DOF.

As to C_{pos}, the constraints are that the object can change the position only to the adjacent cells and the robots cannot change their relative position to the object (the position change operation). The individual robot motions are planned as straight-line trajectories to the next cell, Fig. 10(a).

As to C_{ori}, the constraints are that the object can change the orientation only in the present cell and the object and the robots cannot change their position (the orientation change operation). The individual robot motions are planned as circle trajectories around the object's reference point, Fig. 10(b). C_{ori} is discretized in a certain resolution. When the number of the division is D_{ori}, the resolution of C_{ori} is $2\pi/D_{\text{ori}}$ radian and the object's orientation can change in every $2\pi/D_{\text{ori}}$ rad.

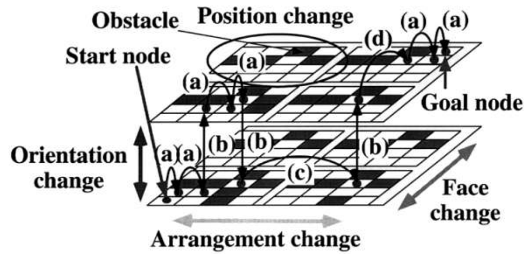


Fig. 11. Discrete representation of the reconstructed C-space. Graph is constructed by nodes and arcs.

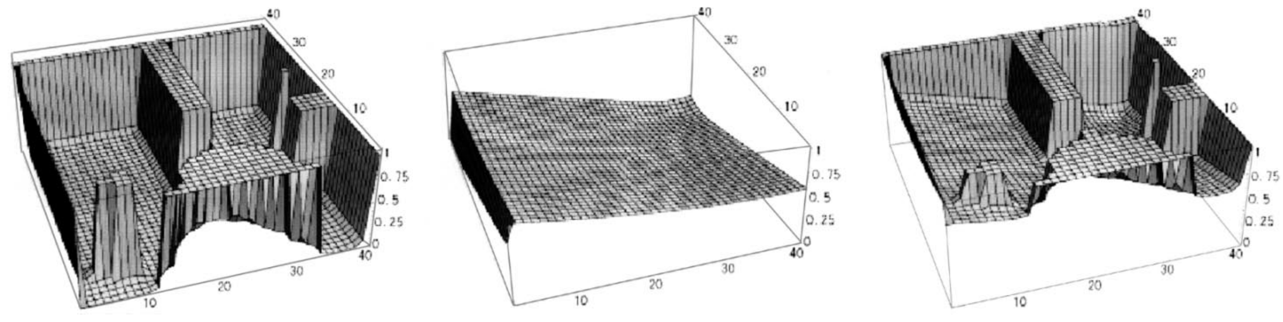
As to C_{arr}, the constraints are that the robots can change their relative position (arrangement) to the object and the object cannot change its position or orientation (the arrangement change operation). The individual robot motions are planned straight-line trajectories along the object's surface, Fig. 10(c). θ_i, Fig. 6(b), is discretized in a certain resolution. When the number of divisions of each robot angle θ_i is D_{arr}, the resolution of θ_i is $2\pi/D_{\text{arr}}$, and the number of state becomes ${}_{D_{\text{arr}}}C_r$ (r: the number of robots). D_{arr} is set not to collide with each robot.

The time to realize each operation is calculated by the length of the trajectory and the velocity of the robots. These two variables are used in the A^* search as costs (Section IV-D). The sweep areas of the robots and the objects are expressed by an octree method, and collision checks are done between the obstacles and the sweep areas.

As to C_{face}, we discretize C_{face} by using $M = (m_{ij})$, as mentioned in Section IV-B. When $m_{ij} = 1$, these configurations are adjacent. The C_{face} dimension of the reduced C-space is coupled with the location of the object. The reference point of the object (v_x, v_y) changes after the face change operation, and the changed v_x and v_y are computed in the local manipulation planner. The changed values are used for planning afterward. The constraints and the individual robots are planned in the local manipulation planner (Section V). Collision checks are also done (Section V-G).

4) Graph Representation: After a discrete representation of the reconstructed C-space is done, the graph constructed by nodes and arcs can be generated (Fig. 11). One node represents one discrete configuration $(v_x, v_y, v_\gamma, v_a, v_f)$. The arcs between two nodes that are adjacent in the discrete C-space C_{pos}, C_{ori}, C_{arr}, or C_{face} are generated. Only one between C_{pos}, C_{ori}, C_{arr}, and C_{face} can change when the present state goes to the adjacent node. This means that one arc means four primitive operations. One arc means the position change operation [(a) in Fig. 11], the orientation change operation [(b) in Fig. 11], the arrangement change operation [(c) in Fig. 11], or the face change operation [(d) in Fig. 11]. We can find a collision-free path from a start node to a goal node with a graph search.

The above can be explained as follows: basically and actually, the path of the object is planned in a 3-D C-space consisting of the object's position (2DOF, (a) in Fig. 11) and the object's orientation about the perpendicular axis to the ground (1DOF, (b) in Fig. 11). We consider the 2-D model, in which the appearance of the object's shape (including robots) changes when the object's face that touches the ground (1DOF) and the robot ar-



(a) (b) (c)

Fig. 12. Potential field. (a) P_{rep}. (b) P_{att}. (c) P_w.

rangement change (1DOF) and the model moves in the previously mentioned 3DOF C-space.

D. Graph Search Method

We adopt the A^* search as a graph search method. The A^* search is a good heuristic search method that can find an optimal solution efficiently if the heuristic function is admissible and monotonic. In our method, we find the path that is free from obstacles and the collision check is needed between the nodes in the discretized graph. As all things are represented in the same way (cells by an octree method), a collision check between two things can be easily done by checking the overlaps of cells, Fig. 9(d). Note that, in this method, the path planning is executed in C-space. However, a C-Obstacle does not compute positively, and the planner checks the collisions in the real world space (not in C-space). This is because the computation costs become larger to calculate a C-Obstacle than to check collisions in real space.

A graph search is performed by using the following evaluation function f

$$f(n) = w_g g(n) + w_h h(n) \tag{9}$$

where

$g(n)$	cost function from the start node to node n;
$h(n)$	heuristic (estimated cost) function from node n to the goal node;
w_g	weight coefficient of a cost function $g(n)$;
w_h	weight coefficient of a heuristic function $h(n)$.

The weight coefficient of a cost function w_g is set as 1 or 0. When $w_g = 1$, we can find an optimal path. When $w_g = 0$, we can find a path quickly but cannot find an optimal path. When $w_h = 0$, the graph search method is the same as Dijkstra's search. When $w_h/w_g > 1$, the heuristic function becomes nonadmissible, but the graph search may be faster if optimality is sacrificed.

The cost function $g(n)$ consists of $g_{\text{time}}(n)$ and $g_{\text{danger}}(n)$. The cost function $g_{\text{time}}(n)$ means the time to transport the object from the start node to the node n. The cost function $g_{\text{danger}}(n)$ means the danger to collide with obstacles [38].

$$\begin{aligned} g(n) =& g_{\text{time}}(n) + w_d g_{\text{danger}}(n) \\ =& g_{\text{pos}}(n) + g_{\text{ori}}(n) + g_{\text{arr}}(n) + g_{\text{face}}(n) \\ &+ w_d g_{\text{danger}}(n) \end{aligned} \tag{10}$$

where

$g_{\text{pos}}(n)$	time to realize the position change operation;
$g_{\text{ori}}(n)$	time to realize the orientation change operation;
$g_{\text{arr}}(n)$	time to realize the arrangement change operation;
$g_{\text{face}}(n)$	time to realize the face change operation;
w_d	weight coefficient of $g_{\text{danger}}(n)$.

The cost function $g_{\text{danger}}(n)$ is calculated using the distance d from the nearest obstacle to the robots and the object

$$g_{\text{danger}}(n) = \frac{1}{d}. \tag{11}$$

After the cost function g is defined in (10), the A^* search is used to find the path in the C-space from the start configuration to the goal configuration to minimize (10). By using this function, we can gain a path that is far from obstacles, and the total time in which robots accomplish the transportation task is short.

The heuristic function $h(n)$ is defined in (12).

$$h(n) = h_{\text{pos}}(n) + h_{\text{ori}}(n) + h_{\text{arr}}(n) + h_{\text{face}}(n) + h_{\text{danger}}(n) \tag{12}$$

where

$h_{\text{pos}}(n)$	estimated time of the position change operation to reach the goal;
$h_{\text{ori}}(n)$	estimated time of the orientation change operation to reach the goal;
$h_{\text{arr}}(n)$	estimated time of the arrangement change operation to reach the goal;
$h_{\text{face}}(n)$	estimated time of the face change operation to reach the goal;
$h_{\text{danger}}(n)$	estimated danger to reach the goal.

We can estimate h_{ori}, h_{arr}, and h_{face} easily by comparing the state at the present node n with the goal node. These three heuristic functions are admissible because the estimated number of these operations does not always exceed the actual number. h_{pos} can be easily chosen to be the straight line distance to the goal that is a common choice in motion planning algorithms, where the C-space is similar to the workspace. However, h_{pos} and h_{danger} cannot be separated because h_{danger} is related strongly to the path. Moreover, h_{danger} cannot be estimated easily because a path becomes complicated under concave obstacles. Therefore, we use a potential function to estimate the distance from the goal and the danger cost together. h_{pos} and h_{danger} are expressed as a potential function.

A potential field is constructed with a repulsive force from obstacles and an attractive force from a goal configuration (Fig. 12).

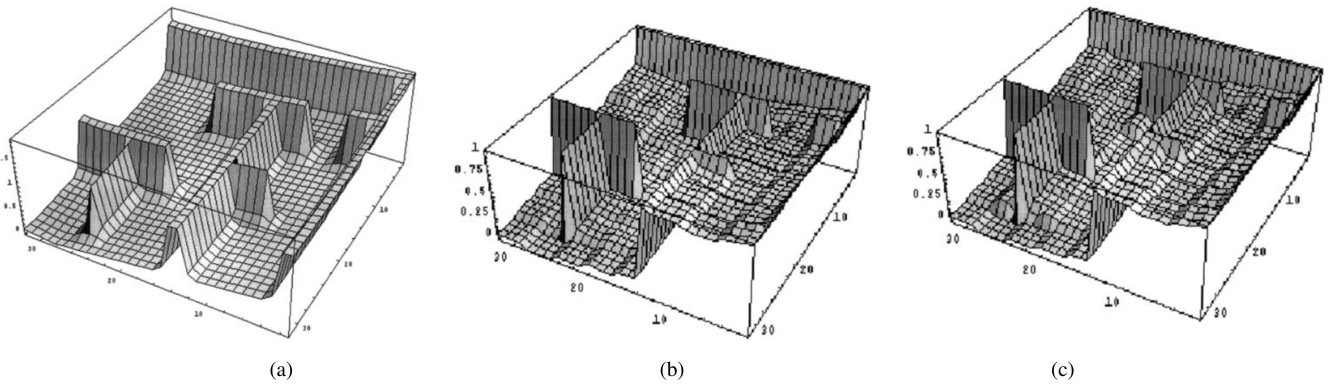


Fig. 13. Three potential fields. (a) Euclid potential field. (b) Wavefront potential field. (c) Skeleton wavefront potential field.

The total force from the potential field, where the configuration of the object and robots is x_j, can be calculated as a sum of the force acted on the object cell $A_{\mathrm{obj},j}$ ($j = 1, \ldots, k$), see Fig. 9(c). The total force acted on x_j is defined in (13), and h_{pos} and h_{danger} are defined in (14) (c is the number of the object cell). In (14), we sum up the potential over the cells on the path found from the potential field.

$$P_w(A_{\mathrm{obj},j}) = P_{\mathrm{att}}(x_j) + P_{\mathrm{rep}}(x_j) \tag{13}$$

$$h_{\mathrm{pos}}(n) + h_{\mathrm{danger}}(n) = \sum_{i=1}^{c} P_w(A_{\mathrm{obj},i}(n)) \tag{14}$$

where

x_j	reference point of object cell $C_{\mathrm{obj},j}$;
$P_w(A_{\mathrm{obj},j})$	force acted on the object and robots;
$P_{\mathrm{rep}}(x_j)$	repulsive force from obstacles;
$P_{\mathrm{att}}(x_j)$	attractive force from the goal.

In this paper, we adopt three potential fields: (A) Euclid potential, (B) Wavefront potential, and (C) Skeleton wavefront potential. As to the Euclid potential field, we adopt the potential field mentioned in [39], which is one of the simplest potential functions. Concerning the wavefront potential field, we use the concept of the wavefront expansion mentioned in [3]. As to the skeleton wavefront potential field, we generate the potential fields with the skeleton method and wavefront expansion [8]. The potential fields are generated in the real world space and not in C-space to reduce the computation costs, as in the octree mentioned in IV-C.3 (The path planning is executed in C-space).

The three potential fields are shown in Fig. 13, and this potential function is normalized with the velocity of the robots. Therefore, the dimension of the potential function is the same as the dimension of the time. In order to set the repulsive force from obstacles to be smaller than d and the attractive force from the goal to be smaller than the time of the real robot motion speed, $h_{\mathrm{pos}} + h_{\mathrm{danger}}$ becomes admissible.

The evaluation function f defined in (9) is not always the heuristic function that has monotonicity [40]. Therefore, we can make a monotonic heuristics with a pathmax

$$f(n') = \max\Big(f(n), g(n') + h(n')\Big) \tag{15}$$

where n' is a child node of n.

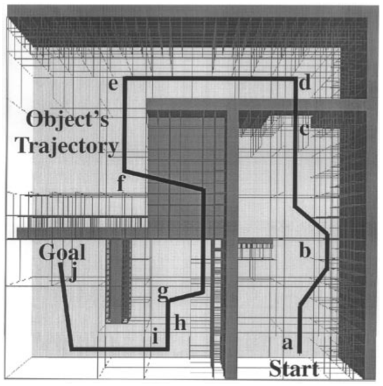


Fig. 14. Simulation result. A trajectory of the object from the start position to the goal position.

The heuristic function used in this search is not necessarily admissible. However, this search method can efficiently find appropriate solutions within a practical time.

E. Simulation

We verified the global path planner through the simulation. The motion planner computes the paths in which two robots transport an L-shape large object in a 3-D environment.

Simulation results are shown in Figs. 14 and 15. The positions (a) – (j) shown in Fig. 14 correspond to Fig. 15(a)–(j). When the robots pass through a narrow space (for example, $a \to c, c \to d$, or $f \to g$), the robots execute the orientation change operations and the arrangement change operations. When the robots must pass through a very narrow space (for example, $h \to j$), the robots execute the face change operation in a wide space ($g \to h$). The manipulation way is planned with the local manipulation planner mentioned in the next section.

The result of simulations shows that the global path planner can generate collision-free paths of the robots and object in a certain complicated environment where the number of obstacles is over 10 and includes hollow objects and some other complex situations [39].

However, in a specially complicated case (for example, the door is too narrow for robots to handle the object's complicated shape, and the robot must go through the door without handling it), this planner cannot find a solution because the robots never move away from the object in the position change operation, in

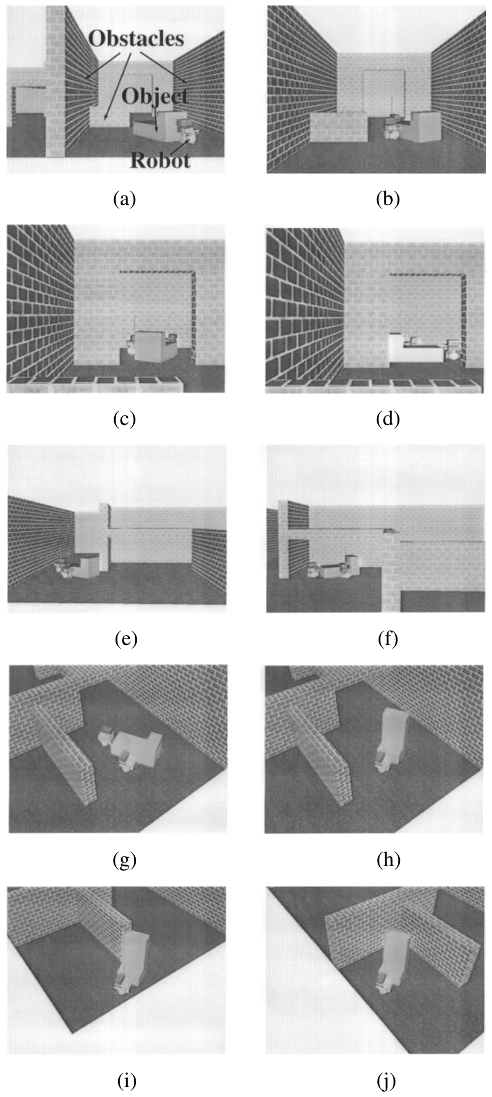


Fig. 15. Simulation result. Overview of the robots and the object in each step.

the orientation change operation, or in the arrangement change operation.

Qualitative analysis is carried out in Section VI.

V. Local Manipulation Planning

A. Outline of Local Planner

In the local manipulation planner, the motions of the robots and object are planned while they manipulate it (the face change operation). At first, we formulate the manipulation under the assumption that the object moves quasi-statically (Section V-B). From the result of the analysis, we make *a stable domain graph* that indicates the position of the stick and the angle of the object when the object is stable (Section V-C). In the next step, we select characteristic points at which the operation method may change in the stable domain graph and make *an operation graph* that indicates the discretized object's motion (Section V-D). After searching the manipulation way in the operation graph, the actual robot motions are planned (Section V-E).

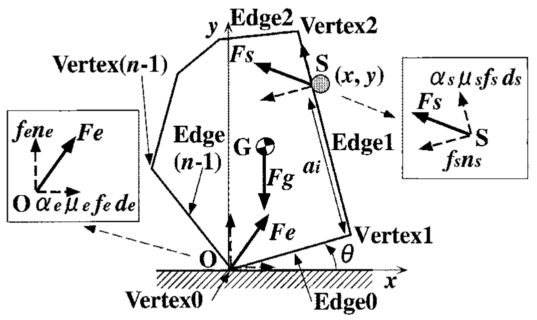


Fig. 16. 2-D model of an object. The formulation is done according to the quasi-static analysis.

B. Formulation of Manipulation

Under the assumptions in Section III-E and the situation mentioned in Fig. 16, a 2-D model can express this situation in the local manipulation planner. We can describe these constraints about the object

$$F_s + F_e + F_g = 0 \tag{16}$$

$$P_s \times F_s + P_g \times F_g = 0 \tag{17}$$

where

F_s		force at stick point S;
F_e		force at contact point O;
G	$= (0, -mg)$	force at the center of the mass;
P_s	$= (x, y)$	position of the stick;
P_g	$= (x_g, y_g)$	center of the mass of the object.

Let f_s be the normal contact force at S, and let f_e be the normal contact force at O

$$F_e = f_e n_e + \alpha_e \mu_e f_e d_e \tag{18}$$

$$F_s = f_s n_s + \alpha_s \mu_s f_s d_s \tag{19}$$

where $n_s = (u_n, v_n)$ and $d_s = (u_d, v_d)$.

From (16) to (19), we obtain the following equation:

$$f_s u_n + \alpha_s \mu_s f_s u_d + \alpha_e \mu_e f_e = 0 \tag{20}$$

$$f_s v_n + \alpha_s \mu_s f_s v_d + f_e - mg = 0 \tag{21}$$

$$f_s(x v_n - y u_n) + \alpha_s \mu_s f_s (x v_d - y u_d) - mg(x_g \cos\theta - y_g \sin\theta) = 0 \tag{22}$$

where

f_s	force at S;
f_e	force at O;
$\mu_{s,e}$	coefficient of friction at S, O;
m	mass of the object;
θ	angle of the object.

We can know whether the object is stable or not by solving (20)–(22) under the constraint of inequalities (23)–(27)

$$f_e \geq 0 \tag{23}$$

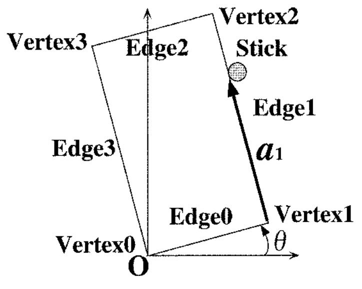
$$f_s \geq 0 \tag{24}$$

$$|\alpha_e| \leq 1 \tag{25}$$

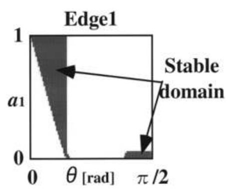
$$|\alpha_s| \leq 1 \tag{26}$$

$$f_s \leq F_{\max} \tag{27}$$

where $F_{\max}$ is the maximum force of a robot.

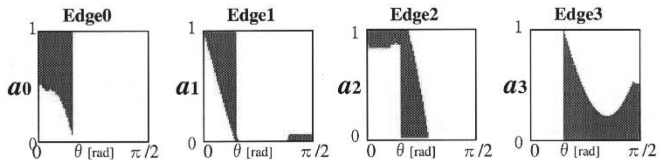


(a)

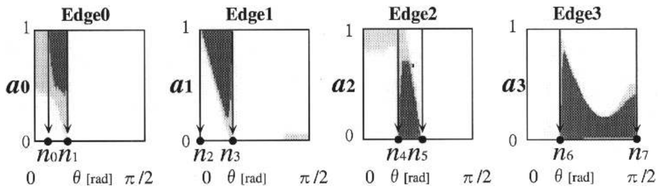


(b)

Fig. 17. Stable domain of Edge 1. (a) Parameter a_i and θ. (b) Stable domain graph.



(a)



(b)

Fig. 18. Stable domain graph of each edge. (a) Stable domain graph without errors. (b) Stable domain graph with errors.

Inequalities (23)–(24) mean that the object is always in contact with the stick and the floor. Inequalities (25)–(26) mean that the object does not slip and remains stable. Inequality (27) means that the robots do not apply force to the object beyond their ability.

C. Stable Domain Graph

We make a stable domain graph that indicates the position of the stick and the angle of the object when the object is stable. The Y axis means that parameter a_i shows the contact position with the object, and the X axis means that parameter a_i shows the angle of the object (Fig. 17). The parameter a_i ($0 \leq a_i \leq 1$) expresses the contact point of the stick at Edge i of the object. For example, when $a_1 = 0$, the stick contacts the object at Vertex 1. When $a_1 = 0.5$, the stick contacts the object at the center of Edge 1.

The object is stable if the parameter (θ, a_i) is in a stable domain. Therefore, if the object is operated in this domain, the object manipulation will not fail. Because the number of edges is four in the case of rectangles, four stable domain graphs, which show a stable domain for each edge, are generated, Fig. 18(a). In addition, the limitation on the movement of robots (for example, the end-effectors' height, which is limited from $h_{\min}$ to $h_{\max}$), can be considered in the stable domain graph.

The effects of motion errors of the mobile robots and environmental indefiniteness are taken into consideration. The error factors that influence a stable domain are (i) the motion errors of mobile robots, and (ii) the accuracy of given data and the change of a coefficient of friction. Concerning the former problem, we search for a domain where the stability of the object is maintained if a position error ($\pm\Delta x$ or $\pm\Delta y$) exists on the position of the stick. We can put the bend of the stick on this position error. As to the latter problem, we search for a domain where the stability of the object is maintained if the coefficient of friction $\mu_{s,e}$ changes within the range of $\pm\Delta\mu_{s,e}$. The input data errors (position of the center of the mass ($\pm\Delta x_g$, $\pm\Delta y_g$), mass of the object $\pm\Delta m$) can be considered. The stable domain graph considering the motion errors is shown in Fig. 18(b). In Fig. 18(b), the domain is reduced compared with Fig. 18(a). In the domain of difference between the domain in Fig. 18(a) and (b), work fails when there are motion errors or another factors.

D. Operation Graph

We generate the operation graph and utilize it to plan the motions of the robots efficiently. The procedure of generating the operation graph is as follows. First, we select characteristic points where the operation method may change in the stable domain graph. Concretely, we choose the points whose θ values are the largest and the smallest in a stable domain, Fig. 18(b). These points are regarded as nodes in the operation graph. Therefore, in the operation graph, all nodes that have physical meanings are collected, Fig. 19(a). The information that the nodes hold in the operation graph consists of the edge number and the angle of the object θ. In the operation graph, information about a_i is not considered to plan the motions of the sticks efficiently.

In the operation graph, the pose change of the object is planned with movement from one node to another. In the rectangle case shown in Fig. 17(a), the pose change is performed by moving from node n_2 (0 rad) to node n_7 ($\pi/2$ rad).

In the operation graph, the arc between two nodes indicates three types of operation methods, i.e., (a) the continuous operation, (b) the hand-over operation, and (c) the transfer operation. The arc that is a horizontal line between two nodes indicates the continuous operation, Fig. 19(a). In this operation, it is possible to change the angle of the object without changing the edge that the stick contacts. During the continuous operation, the contact point of the stick continuously changes. This means that a_i in the stable domain graph changes continuously. The perpendicular arc indicates the hand-over operation, Fig. 19(b). In this operation, it is possible to change the edge that the stick contacts without changing the angle of the object. Nodes n_8 and n_9 are newly generated in the operation graph. The oblique arc indicates the transfer operation, Fig. 19(c). In this operation, it is possible to change both the angle of the object and the edge that the stick contacts. This operation is performed when there is no stable domain between two nodes.

To express the difficulties of these operations, the distance of the arc, which means the moving cost between two nodes, is introduced. Then, the problem of choosing the manipulation

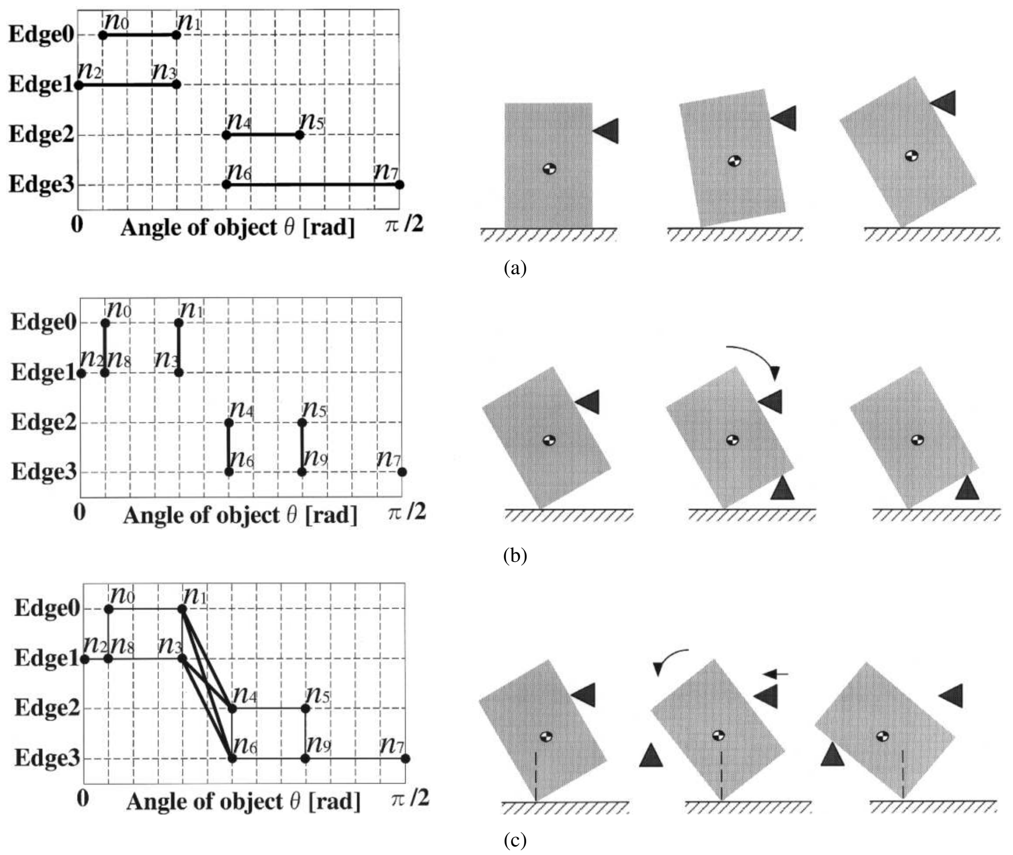

Edge0
Edge1
Edge2
Edge3
n_0 n_1 n_2 n_3 n_4 n_5 n_6 n_7 n_8 n_9
0
Angle of object θ [rad]
$\pi/2$
(a)
(b)
(c)

Fig. 19. Operation graphs and operations. (a) Continuous operation. (b) Hand-over operation. (c) Transfer operation.

method among three operations goes back to the problem of finding the shortest path. A long distance from the start node to the goal node means that the pose changing operation of the object is difficult. Therefore, we choose the shortest path because the pose-changing task can be performed easily. The distance (cost) of each arc is defined in (28) ($\Delta\theta$ means the change of the object angle). In our planner, we set $A_a = 1$, $A_b = 10^2$, $A_c = 10^6$, $B_a = 0$, $B_b = 10^2$, and $B_c = 10^{-1}$.

$$cost_{a,b,c} = A_{a,b,c}(\Delta\theta + B_{a,b,c}). \tag{28}$$

Accordingly, the costs are determined as follows: $cost(a) < cost(b) < cost(c)$. The cost of the continuous operation is lower because it is the easiest way to manipulate the object. This operation can be accomplished with one stick, and the total time of operation is the shortest compared to that of other operations. The cost of the transfer operation is higher, on the other hand, because it is the most difficult. This operation needs two sticks, and the object is not stable during this operation. When this operation is performed, a greater effect of the uncertain factor of the environment must be considered compared with other operations.

After the costs of arcs are determined, the A^* search algorithm is used to solve the shortest path-planning problem. From the result of the search, the shortest path is gained; n_2 (initial state) $\rightarrow n_8 \rightarrow n_3 \rightarrow n_6 \rightarrow n_9 \rightarrow n_7$ (goal state). Path $n_2 \rightarrow n_8 \rightarrow n_3$ means the continuous operation on Edge 1. Path $n_3 \rightarrow n_6$ means the transfer operation between the stick on Edge 1 and Edge 3. Path $n_6 \rightarrow n_9 \rightarrow n_7$ means the continuous operation on Edge 3.

The position where the stick comes in contact with each edge (a_i) is not determined. The result obtained here is the order of the operation in which the stick comes into contact with each edge. The orbit of the stick is not determined at present.

E. Determination of Stick Orbit

In this research, the stick orbit is determined after a procedure of object operation is determined. If it is a manipulation method for fixed manipulators, the orbit of the contact position where the force applied to the stick is always minimized may be good. For example, when the stick contacts Edge 1, the continuous operation where a_1 is near 0 leads to the situation in which the control force is minimum.

However, when optimizing the control force, the orbit of a stick becomes nearly circular in shape. In this case, the perpendicular and the horizontal speed of the stick need to be controlled in a complicated way every time. his carries the risk of causing the task to fail, for it is inconvenient for mobile robots to make a complicated operation.

Then, the stick orbit of the continuous operation is such that mobile robots can easily control it. In this research, a straight-line orbit is adopted.

The stick positions when the continuous operation begins and ends are determined to minimize the force at these points, and the meantime is connected linearly.

In the rectangle operation dealt with here, the locus of each stick is, respectively, shown in Fig. 20(a-1), and the orbit is in a stable domain, Fig. 20(a-2). When the orbit is out of the stable domain, the stick positions of the beginning and the end are

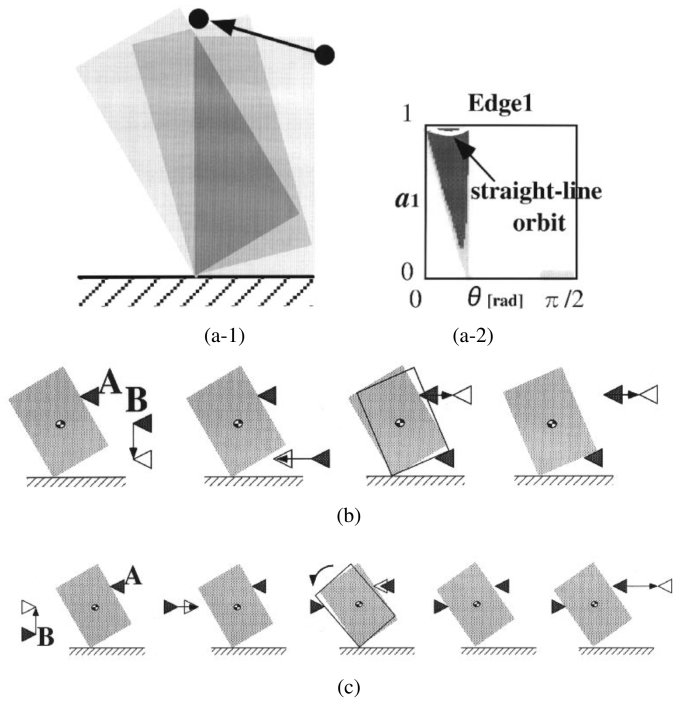


Fig. 20. Decision of stick orbit. (a) Continuous operation. (b) Hand-over operation. (c) Transfer operation.

TABLE I
PARAMETERS FOR SIMULATION OF MANIPULATION

Size of object	0.50m x 1.00m
Mass of object	4.00kg
Friction coefficient	0.20
Mass of gravity	Center of object
Max force of robots	19.6N
[Condition 1]	Without errors
[Condition 2]	Consider errors
Motion errors of robots	(± 0.04m, ± 0.01m)
Mass, friction coefficient	± 10%
Center of mass of object	(± 0.01m, ± 0.01m)
[Condition 3]	With limitation
Movable range of stick height	0m – 0.50m

suitably changed. Then, a solution in which the orbit is always in the stable domain is searched.

The way in which the hand-over operation and the transfer operation are generated is shown in Fig. 20(b) and (c), respectively. The straight-line orbits are also adopted in these operations.

F. Simulations

Table I shows parameters for simulations of an object manipulation. The manipulation method of the object shown in Fig. 21 is planned off-line by the local manipulation planner.

In Condition 1, the motion errors of robots are not considered. Therefore, a dangerous manipulation method is planned, and the robots contact the vertexes, Fig. 21(a). If there are very few motion errors, the manipulation fails. In Condition 2, the motion errors of robots are considered. The manipulation method is robust for the motion errors, Fig. 21(b), because the robots contact

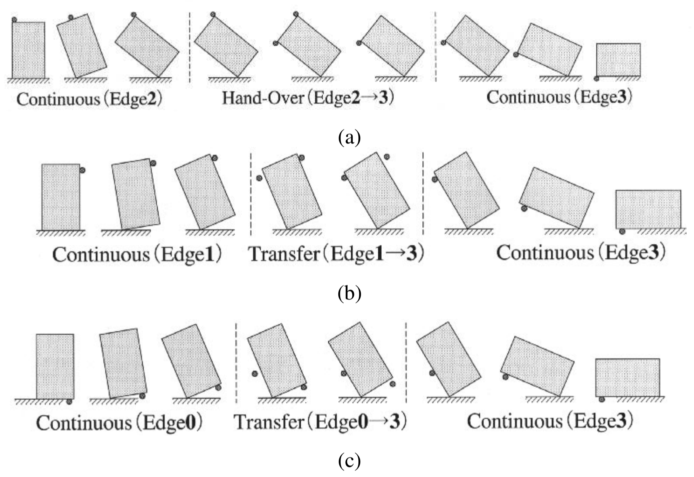


Fig. 21. Planned manipulation method. (a) Without errors. (b) With errors. (c) With limitation of stick movable range.

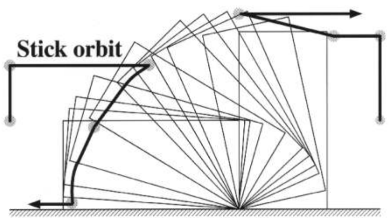


Fig. 22. Orbit of sticks in the case (planning result with motion errors) of Fig. 21(b).

the edges with margins from the vertex. The orbit of sticks in this case (planning result with motion errors) is shown in Fig. 22.

In Condition 3, the limitations of the movable range of the sticks and the motion errors of robots are considered, Fig. 21(c). When the stick cannot move to a high position, the robots touch at low positions and realize the manipulation.

The results of simulations show that our proposed local manipulation planner can cope with various situations.

G. Manipulation Cost and Manipulation Space

The manipulation space where robots work with an object is represented by the cell decomposition method (the octree method). The manipulation cost, which means the total time to finish the manipulation, is calculated in the local manipulation planner. The information is used in the global path planner.

VI. Evaluation of the Motion Planner

We verified the proposed planner in this paper through a simulation. After a few experiments, we picked the skeleton wavefront potential and the coefficients for the A^* search in global motion planning. When the environment becomes complicated and the distance between the start and the goal becomes large, the effectiveness of the skeleton wavefront potential stands out, and the number of the open node N_{open} becomes the smallest (about one fourth smaller in comparison with the Dijkstra's algorithm using the Euclid potential).

The motion planner computes the paths in which two robots transport an L-shape large object in a 3-D environment. Simulation results of the global path planner and the local manip-

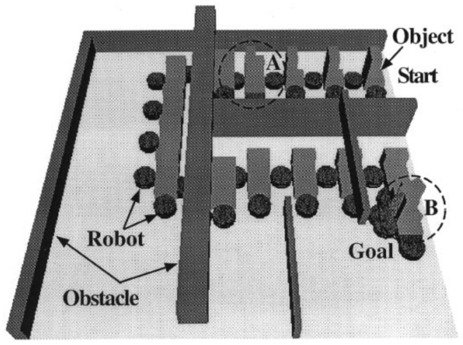


Fig. 23. Simulation results. Result of global path planning.

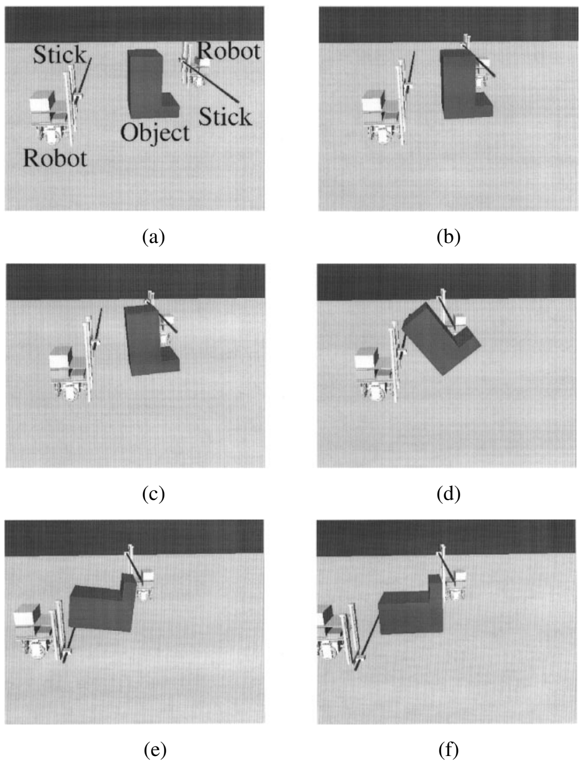


Fig. 24. Simulation results. Result of local manipulation planning at point A in Fig. 23.

ulation planner are shown in Figs. 23 and 24, respectively. (In Fig. 24, obstacles are not displayed because of the simplicity of observation). In Fig. 23, the robots transport the object while avoiding obstacles, and they change the pose of the object at points A and B. The number of manipulation times can be reduced. In Fig. 24, the manipulation way of the object at point A is computed. In this simulation, an objective task has been realized without failure, i.e., the object did not slip or fall down due to motion errors of the robots and environmental indefiniteness.

It takes about 3000 CPU s to compute the path of the object and robots with Ultra SPARC-II (334 MHz) in the global path planning. In this simulation condition, N_{open} is about 2500, and it takes about 600 CPU s to compute one manipulation way in the local manipulation planner. The whole computation time is about 6000 CPU s. The environment is 12.8 m $\times$ 12.8 m $\times$ 12.8 m. The resolution of the global path planning is 0.1 m, and that of the local manipulation planning is 0.01 m and 1.0°.

This shows that, despite the complexity of the problem, our proposed planner can efficiently find appropriate solutions within a practical time.

VII. Conclusion

In this paper, we proposed a motion planning method for cooperative transportation of a large object by multiple mobile robots in a 3-D space. We divided the motion planner into a local manipulation planner and global path planner.

For the global motion planner, we reduced the dimensions of the C-space and could find a solution by searching in this smaller dimensional C-space using the potential function. For the local manipulation planner, we considered the motion errors in a planning stage beforehand and built a manipulation technique that is suitable for position-controlled mobile robots. After constructing the two planners, we integrated them. In other words, a gross motion planner and a fine motion planner could be integrated by our method.

Simulations have verified the effectiveness of the proposed planning method.

Acknowledgment

The authors would like to thank the research group members M. Fukuchi and K. Kawano for their help.

References

[1] T. Arai and J. Ota, “Let us work together-Task planning of multiple mobile robots,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1996, pp. 298–303.
[2] H. Asama, M. Sato, L. Bogoni, H. Kaetsu, A. Matsumoto, and I. Endo, “Development of an omnidirectional mobile robot with 3DOF decoupling drive mechanism,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 1995, pp. 1925–1930.
[3] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.
[4] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.
[5] Y. K. Hwang and N. Ahuja, “Gross motion planning–A survey,” *ACM Comput. Surveys*, vol. 24, no. 3, pp. 219–292, 1992.
[6] K. Gupta and A. P. Del Pobil, Eds., *Practical Motion Planning in Robotics*. New York: Wiley, 1998.
[7] K. Kondo, “Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration,” *IEEE Trans. Robot. Automat.*, vol. 7, pp. 267–277, June 1991.
[8] J. Barraquand and J.-C. Latombe, “Robot motion planning: A distributed representation approach,” *Int. J. Robot. Res.*, vol. 10, no. 6, pp. 628–649, 1991.
[9] Y. Koga and J.-C. Latombe, “On multi-arm manipulation planning,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 1994, pp. 58–63.
[10] K. K. Gupta and Z. Guo, “Motion planning for many degrees of freedom: Sequential search with backtracking,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 1992, pp. 2328–2333.
[11] Y. K. Hwang and P. C. Chen, “A heuristic and complete planner for the classical mover’s problem,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 1995, pp. 729–736.
[12] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Automat.*, vol. 12, pp. 566–580, Aug. 1996.
[13] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, “Analysis of probabilistic roadmaps for path planning,” *IEEE Trans. Robot. Automat.*, vol. 14, pp. 166–171, Feb. 1998.

[14] P. C. Chen and Y. K. Hwang, "SANDROS: A dynamic graph search algorithm for motion planning," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 390–403, June 1998.
[15] H. Terasaki and T. Hasegawa, "Motion planning of intelligent manipulation by a parallel two-fingered gripper equipped with a simple rotating mechanism," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 207–219, June 1998.
[16] M. Erdmann, "An exploration of nonprehensile two-palm manipulation," *Int. J. Robot. Res.*, vol. 17, no. 5, pp. 485–503, 1998.
[17] K. M. Lynch, "Toppling manipulation," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1999, pp. 2551–2557.
[18] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1995, pp. 235–242.
[19] M. Erdmann, "Using backprojections for fine motion planning with uncertainty," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 19–45, 1986.
[20] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 53–71, 1986.
[21] R. C. Brost, "Automatic grasp planning in the presence of uncertainty," *Int. J. Robot. Res.*, vol. 7, no. 7, pp. 3–17, 1988.
[22] M. A. Peshkin and A. C. Sanderson, "The motion of a pushed, sliding workpiece," *IEEE J. Robot. Automat.*, vol. 4, pp. 569–598, Dec. 1988.
[23] S. Akella and M. T. Mason, "Posing polygonal objects in the plane by pushing," *Int. J. Robot. Res.*, vol. 17, no. 1, pp. 70–88, 1998.
[24] M. Hashimoto, F. Oba, and S. Zenitani, "Object-transportation control by multiple wheeled vehicle-planar cartesian manipulator systems," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1995, pp. 2267–2272.
[25] T. G. Sugar and V. Kumar, "Control of cooperating mobile manipulators," *IEEE Trans. Robot. Automat.*, vol. 18, pp. 94–103, Feb. 2002.
[26] K. Kosuge, T. Oosumi, and K. Chiba, "Load sharing of decentralized-controlled multiple mobile robots handling a single object," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1997, pp. 3373–3378.
[27] O. Khatib, "Mobile manipulation: The robotic assistant," *Robot. Auton. Syst.*, vol. 26, pp. 175–183, 1999.
[28] N. Sawasaki and H. Inoue, "Cooperative manipulation by autonomous intelligent robots," *JSME Int. J.*, ser. C, vol. 39, no. 2, pp. 286–293, 1996.
[29] J. Ota, N. Miyata, T. Arai, E. Yoshida, D. Kurabayashi, and J. Sasaki, "Transferring and regrasping a large object by cooperation of multiple mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1995, pp. 543–548.
[30] H. Osumi, "Cooperative strategy for multiple mobile manipulators," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1996, pp. 554–559.
[31] M. Hara, M. Fukuda, H. Nishibayashi, Y. Aiyama, J. Ota, and T. Arai, "Motion control of cooperative transportation system by quadruped robots based on vibration model in walking," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1999, pp. 1651–1656.
[32] Y. Aiyama, M. Hara, T. Yabuki, J. Ota, and T. Arai, "Cooperative transportation by two four-legged robots with implicit communication," *Robot. Auton. Syst.*, vol. 29, pp. 13–19, 1999.
[33] C.-K. Yap, "How to move a chair through a door," *IEEE J. Robot. Automat.*, vol. RA-3, pp. 172–181, June 1987.
[34] A. Yamashita, K. Kawano, J. Ota, T. Arai, M. Fukuchi, J. Sasaki, and Y. Aiyama, "Planning method for cooperative manipulation by multiple mobile robots using tools with motion errors," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1999, pp. 978–983.
[35] A. Yamashita, M. Fukuchi, J. Ota, T. Arai, and H. Asama, "Motion planning for cooperative transportation of a large object by multiple mobile robots in a 3D environment," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2000, pp. 3144–3151.
[36] Y. Aiyama, M. Inaba, and H. Inoue, "Pivoting: A new method of graspless manipulation of object by robot fingers," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1993, pp. 136–143.
[37] A. Yamashita, J. Sasaki, J. Ota, and T. Arai, "Cooperative manipulation of objects by multiple mobile robots with tools," in *Proc. 4th Japan-France/2nd Asia-Europe Congr. Mechatronics*, 1998, pp. 310–315.
[38] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE J. Robot. Automat.*, vol. RA–2, pp. 135–145, June 1986.
[39] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 23–32, Feb. 1992.
[40] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

Atsushi Yamashita (S'00–A'02–M'02) received the M.Eng. and Ph.D. degrees in engineering from the Department of Precision Engineering, University of Tokyo, Tokyo, Japan, in 1998 and 2001, respectively.

From 1998 to 2001, he was a Junior Research Associate in the RIKEN (Institute of Physical and Chemical Research). He has been a Research Associate in the Department of Mechanical Engineering, Shizuoka University, Shizuoka, Japan, since 2001. His research interests are motion planning of multiple mobile robots, mechanism of omni-directional mobile robots, and robot vision.

Dr. Yamashita is a member of the IEEE Robotics and Automation Society.

Tamio Arai (M'92) received the M.Eng. and Dr.Eng. degrees in engineering from the Department of Precision Engineering, University of Tokyo, Tokyo, Japan, in 1972 and 1977, respectively.

He was a Visiting Researcher in the Department of Artificial Intelligence, Edinburgh University, Edinburgh, U.K., from 1979 to 1981. He has been a Professor in the Department of Precision Engineering, University of Tokyo, since 1987. His specialties are assembly and robotics, especially multiple mobile robots, including the legged robot league of RoboCup. He works in IMS programs in Holonic Manufacturing System. He has contributed in robot software in ISO activity. Since 2000, he has been a director of RACE (Research into Artifacts, Center for Engineering), the University of Tokyo, and proposes a new field of engineering, Service Engineering.

Dr. Arai is an active member of CIRP, a member of the Robotics Society of Japan, Japan Society for Precision Engineering, and Honorary President of the Japan Association for Automation Advancement.

Jun Ota (M'92) received the M.Eng. and Ph.D. degrees in engineering from the Department of Precision Engineering, University of Tokyo, Tokyo, Japan, in 1989 and 1994, respectively.

From 1989 to 1991, he was with the Nippon Steel Cooperation. In 1991, he was a Research Associate of the University of Tokyo. In 1996, he became an Associate Professor at the Graduate School of Engineering, the University of Tokyo. From 1996 to 1997, he was a Visiting Scholar at Stanford University, Stanford, CA. His research interests are multiple mobile robot systems, environmental design for robot systems, human-robot interface, and cooperative control of multiple robots.

Hajime Asama (M'87) received the M.Eng. and Dr.Eng. degrees in engineering from the Department of Precision Engineering, University of Tokyo, Tokyo, Japan, in 1984 and 1989, respectively.

In 1986, he joined the RIKEN (Institute of Physical and Chemical Research), and became the Professor of RACE (Research into Artifacts, Center for Engineering), University of Tokyo. He served an editorship of *Distributed Autonomous Robotics Systems* (Tokyo: Springer-Verlag), the second and fifth volumes, which were published in 1994, 1996, and 2002, respectively. His main interests are distributed autonomous robotic systems, cooperation of multiple autonomous mobile robots, emergent robotic systems, and intelligent data carrier systems.

Dr. Asama received the JSME Robotics and Mechatronics Division Robotics and Mechatronics Award in 1995, the JSME Robotics and Mechatronics Division Robotics and Mechatronics Academic Achievement Award in 2000, etc. He also received the RoboCup'97 Engineering Challenge Award and RoboCup'98 Japan Open JSAI award as a member of UTTORI United Team. He is a member of RSJ, JSME, SICE, and the New York Academy of Science, etc.