41

Cooperative Transport by Multiple Mobile Robots in Unknown Static Environments Associated With Real-Time Task Assignment

Natsuki Miyata, *Member, IEEE*, Jun Ota, *Member, IEEE*, Tamio Arai, *Member, IEEE*, and Hajime Asama, *Member, IEEE*

***Abstract*—This paper deals with a task-assignment architecture for cooperative transport by multiple mobile robots in an unknown static environment. The architecture should satisfy three features: deal with a variety of tasks in time and space, deal with a large number of tasks compared with the number of robots, and decide behavior in real time. The authors propose the following approach: We consider the unit of task (task instance) as the job that should be done in a short time by one robot. Based on environmental information, task instances are dynamically generated using task templates. The priority of task instances is evaluated dynamically based on the number of robots and the configuration in the workspace. In addition, we avoid generating too many task instances by suppressing object motion. The main part of the architecture consists of two real-time planners: a priority-based task-assignment planner solved by using a linear programming method, and motion planners based on short-time estimation. The effectiveness of the proposed architecture is verified by a cooperative transport simulation in an unknown environment.**

***Index Terms*—Cooperative transport, dynamic task domain, multiple mobile robots, real time, task assignment.**

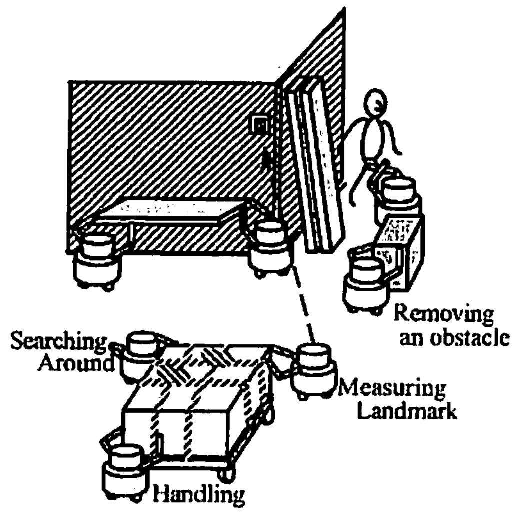

Fig. 1. Cooperative transport in an open environment.

I. Introduction

A SYSTEM for handling flexible materials is in high demand at places such as construction sites and distribution centers. Many researchers (including the authors) have tried to realize such a system by means of the cooperation of multiple mobile robots [1]–[15]. This is because the robots can be distributed freely according to the various types of the objects. The focus of these studies has been on the control and manipulation processes. At places such as the above-mentioned ones, however, robots have to detect changes in the geometry of the surroundings on demand. However, the sensing process takes a considerable amount of time because of the limited ability of each robot's sensor. Hence, to cope with unexpected situations and achieve the final goal of the robot group (to transport a given object to a certain configuration), it is necessary to plan and execute various tasks including the sensing process as well as the manipulating process (Fig. 1). For example, in order to handle an object, robots must recognize their own positions based on known landmarks. They must then search around their surroundings adequately in order to detect unexpected situations as soon as possible. If a robot finds a movable object like a trashcan or a door in its way, it has to change the object's path or remove the obstacles. When robots find a moving obstacle (e.g., a human), they have to approach it and instruct it to get out of their way.

In this paper, these various required actions of robots are called "tasks." When we focus on a task-assignment architecture, there are three parts to consider (Sections I-A.1, I-A.2 and I-A.3). First of all, robots have inadequate information about their working space at the beginning of the whole work. The task will be generated as a result of their action. From the viewpoint of the "characteristics of a task" in such work, the task assignment planner should have the ability to deal with the following two aspects (Section I-A.1 and I-A.2).

1) Dynamic Change of Tasks: There are several kinds of requirements for the robots to achieve their main purpose of moving an object to a certain configuration, and not all of them are always needed. For example, a trashcan in the robots' way

Manuscript received March 12, 2001; revised March 28, 2002. This paper was recommended for publication by Associate Editor L. Parker and Editor S. Hutchinson upon evaluation of the reviewers' comments. This paper was presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyongju, Japan, October, 1999, and at the IEEE International Conference on Robotics and Automation, San Francisco, CA, April 2000.

N. Miyata is with the Digital Human Laboratory, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan (e-mail: n.miyata@aist.go.jp).

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

H. Asama is with the Chemical Engineering Laboratory, Institute of Physical and Chemical Research (RIKEN), Saitama 351-0198, Japan (e-mail: asama@cel.riken.go.jp).

Digital Object Identifier 10.1109/TRA.2002.803464

must be displaced at the time it is detected. That is, tasks change dynamically according to the translation of the robot group. Even though the required kind of task is the same, tasks that arise in a different space are different in quality. For example, searching an area that was previously checked by someone else is different from observing an area that has never been observed. Tasks, therefore, should be quickly abstracted, reflecting the situation that the robots are facing.

2) Too Many Tasks for Robots: Along with the change of the tasks mentioned in Section I-A.1, the total number of task changes sometimes exceeds the amount of work that can be done by the available robots. These tasks should be selected and executed in the appropriate order. When determining an appropriate execution order, it is necessary to consider the inherent characteristics of each task against other different kinds of tasks. It is also necessary to select and execute a cooperation task that can secure an adequate number of robots from the available ones. In addition, it is also important to prevent an overflow of task-instance generation because it will lead to the bankruptcy of the task assignment system.

On the other hand, these tasks should be solved when the robots are moving in the working area. Therefore, from the viewpoint of the planner of robots who move and work, they should be realized.

3) Behavior Planning in Real Time: There are several styles to transfer an object. One simple way is to push the object without connecting the robots and the object. For example, Kimura *et al.* dealt with object manipulation by pushing in space and considering a dynamic manipulability margin [1]. Mataric *et al.* realized a box-pushing task using multiple behavior-based mobile robots [2]. This style of transport is, however, not applicable directly to a general environment and manipulation because of its high dependence on the floor status.

Another style of transport is to use mobile manipulators. Khatib used two mobile manipulators, controlling the inner force between them [3]. Osumi introduced a passive joint into the manipulators to absorb the robots' positioning error [4]. Desai proposed collaborative motion planning for the mobile base and manipulator part [5]. Research for this style of transport mainly focuses on how to coordinate the motion of the base and that of the manipulator under limitations. Though this style of transport has the advantage of dexterous manipulation, the control system becomes complicated.

The rest of the studies utilized wheeled robots with a simple end-effector. Stilwell presented a transport system with many ant-like homogeneous robots [6] and discussed how to distribute the load among them [7]. Hashimoto *et al.* realized the transfer of a palletized object with a master-slave robot system [8]. They also proposed a dynamic control method with couplers between an object and each robot [9]. Kosuge *et al.* presented a method to estimate and control the object motion in a decentralized robot system based on information obtained by each robot's torque sensor [10]. They also extended their method to the cooperation between robots and humans [11]. Wang *et al.* realized the cooperative transport by pushing using a simple mechanism for each robot's arm to avoid too much internal force [12]. The focus of these studies has been on control problems in manipulation, such as methods to control internal forces. In contrast to the above, some investigations (including the authors') have proposed an algorithm to change the robots' distribution in cooperative transport according to the surroundings [13]–[15]. These studies focused on how to reflect information about the surroundings in the planning but not on the process to get it.

As for the task assignment method for multiple mobile robots, numerous trials have been carried out. One approach is to assign a task based on negotiation through communication [16]–[19]. However, this approach takes much time in negotiation and is not appropriate for motion decisions in real time.

Another approach is to assign a task using behavior-based robots [20]–[23]. In the case of the first three, it is difficult to express mutual dependency among tasks in terms of the order (or timing) in which they must be executed. The last one is difficult to deal with the task that requires several robots to move at once. Therefore, these methods are not applicable directly to the real-time cooperative transport system that we are discussing here. The optimal allocation problem has also been studied in the research field of the *Job-shop Scheduling Problem*. However, most of them present difficulties of application to a dynamic task domain with frequent change.

Therefore, the aim of our research is to construct a task-assignment architecture that can be applied to tasks for cooperative transport having the three characteristics described above.

The contents of this paper are as follows. In Section II, we show the profile of the proposed task-assignment architecture. In Section III, the contents of the task template and task assignment algorithm are explained. Section IV describes how the tasks are implemented for simulations and an experiment. The effectiveness of the proposed architecture is verified by cooperative transport simulations in Section V and by an experiment in Section VI. We conclude the paper in Section VII.

In this problem, we assume that the robots can communicate among themselves.

II. Task Assignment

A. Architecture of Task Assignment

Just as surroundings might change, required tasks might also change, as shown in Section I-A.1. Therefore, it is of no use to spend much time on planning optimal motion based on the estimation of detailed behavior. Rather, it is necessary to assign tasks and plan each robot's motion in real time according to the brand new sensory information.

We believe that cooperative transport can be expressed and achieved by limited kinds of tasks. Let us divide a time axis into a short sampling period and consider the unit of task as a detailed action that should be done by one robot in one or several periods of time (we call this a *task instance*).

Once the task instance is assigned, detailed motion can be planned for each task in the motion planners embedded in the architecture. Any algorithm can be used for the motion planner that is already in use, or a newly developed one. Then, the point of the task assignment problem is to determine which task instance to select for each robot. If an assignment candidate can be evaluated appropriately, optimal combinations of each robot and a task instance are derived fast in the form of what we call an "assignment problem."

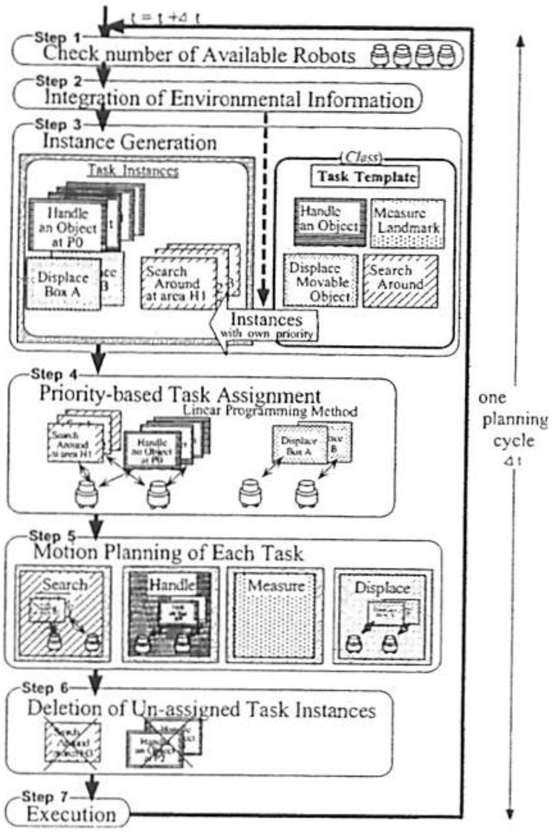

Fig. 2. Architecture of the task-assignment system.

When evaluating candidates, the constant factor and the dynamic factor should be considered. The constant factor is the execution order that is determined by the characteristics of the task. For example, some kind of task should be executed before another task to make sure that the executed results affect the situation properly. Dynamic factors consist of two indices. One is the index that reflects whether there is an adequate number of robots for cooperation. The other is the one calculated based on the time and spatial distribution of task instances and robots.

Accounting for the above, we propose the architecture shown in Fig. 2. This architecture works in the following manner: **Step 1–Step 7** will be repeated at a constant sampling rate.

Step 1) Confirmation of the number of robots that are available for a new assignment.
Step 2) Integration of the environmental information received from each robot.
Step 3) Task-instance generation by inputting environmental information into the task template.
Step 4) Task assignment based on dynamic priority.
Step 5) Motion planning for each task.
Step 6) Deletion of unassigned task instances.
Step 7) Execution by each robot.

The designer should prepare a "task template" and a "motion planner" independently for each task. The former is to generate task instances by estimating the kind and amount of required action in the next sampling cycle (the kinds of tasks, the priority, and the number of robots) based on the environmental information. Details of the task templates are provided in Section III-B.

The latter is to plan the detailed motion of the robots that are assigned to the task during the next sampling cycle. For example, as the motion planner for a task such as "handling an object," we can apply our previous work on transport [15]. Between the motion planners exists a contrivance to suppress the object's translation depending on the progress of other tasks.

As we mentioned at the end of Section I, communication among the robots is basically assumed to be feasible. If the communication from a certain robot fails in **Step 1**, the robot is regarded as not existing. That is, the number of the available robots decreases during that sampling cycle.

B. Problem Solution by the Proposed Method

Here, we explain how this architecture solves the three characteristics described in Section I.

For *dynamic change of tasks*, task instances are generated in each cycle by inputting environmental information into the task templates that are prepared for each task. This enables us to express the dynamic change in the kind and volume of the task.

For *too many tasks for robots*, task-instance generation and assignment are performed with a dynamic calculation of priority based on the inherent execution order and the environmental information. Unassigned instances are deleted once but could be generated afresh and assigned. This means that our architecture offers the mechanism to solve problems sequentially from the basis of the executable ones.

In addition, utilizing the aspect that the tasks arise along with the translation of the robot group, the motion planners suppress the object's translation depending on other tasks' progress. This will work to suppress the generation of too many task instances meaninglessly.

For *behavior planning in real time*, required action is extracted in the form of a task instance that can be performed in a short time by one robot. We also offer the way to calculate each instance's priority. This enables us to formulate the task assignment problem as an "assignment problem." Dividing the time axis into a small period of time also simplifies the motion planning problem by reducing the cost of estimating a complicated motion. Therefore, the planning can be made in real time.

As for the sampling cycle, a shorter sampling cycle is more preferable from the viewpoint of the response of the robot system to the change of the surroundings. On the other hand, each sampling cycle should be long enough to plan how to behave in the next cycle, to transmit this result to all the robots, and each robot to operate its hardware element to achieve the assigned task. It should be determined, therefore, as short as possible, satisfying the constraint to exceed the summation of the following: the time to calculate, the time to communicate, and the time to operate hardware elements to achieve minimum number of tasks.

Continuity and consistency of the physical motion of the robot group is basically considered and guaranteed in the motion planners. In addition to this, a certain kind of task is preferred to be stably assigned to the same robot for a while; that is, we should avoid meaningless switches of the assigned task (a chattering) and a deadlock consequent on the chattering. Our system, therefore, allows two kinds of task instances from the viewpoint of the time to be completed: the one that takes one sampling cycle and the one that takes more than one sampling cycle.

III. Task Template and Assignment Algorithm

A. Task in Assignment

As explained in Section I, we regard various required actions of robots as "tasks." In cooperative transport, typical examples of tasks are the following: *t*1) handling an object; *t*2) recognizing own position by measuring known landmarks; *t*3) searching around; and *t*4) displacing movable objects. In order to execute each task, a robot controls itself to "act" and to "sense." Therefore, these tasks can be classified into three groups according to a robot's "acting" and "sensing" ability: group 1 requires only "acting" (*t*1); group 2 requires only "sensing" (*t*2, *t*3); and group 3 requires both of them (*t*4). This means that if there is no restriction among those groups, a robot can perform a task from group 1 and one from group 2 at the same time.

B. Task Template

Task templates are used to generate task instances by inputting sensory information. Each instance is in the form corresponding to one robot. These instances are assigned to free robots that have finished the instances already assigned to them. In the process of instance generation, therefore, task templates refer to the number of those free robots (n_{freerob}) to determine the total number of instances to generate. n_{freerob} might change every sampling time according to the surroundings. The way to determine each factor of the task template will be explained in Section IV.

C. Assignment Algorithm Using Priority

We introduce integer variables whose value is equal to 1 or 0, according to whether a certain instance is assigned to a certain robot or not. Then, the task assignment problem is formulated as an extended form of what we call "assignment problem" and can be solved in real time. On the other hand, as shown in Section III-A, the required ability of a robot differs from task to task, and sometimes two kinds of tasks are assigned to one robot if those tasks do not conflict in terms of a robot's ability. Thus, we use two variables: S_{ij} for the ability to sense; and A_{ij} for the ability to act. In the case of the task instance that requires both abilities, constraints are added to assign those instances to the same robot. Here, we show the value and meanings of these variables.

$$A_{ij}(\text{or } S_{ij}) = \begin{cases} 1, & \text{if task instance } i \text{ is assigned to} \\ & \text{robot } j\text{'s "Sensing" (or "Acting")} \\ 0, & \text{otherwise} \end{cases}$$

When there exist too many task instances compared to the number of available robots, the system should execute task instances with a higher priority level prior to the ones with lower value. Even if some task instances are in a lower priority level, the assignment system also has to assign as many robots as possible. If the tasks are important, robots must be assigned independently of whether there is an adequate number of robots to cooperatively execute all the instances of the same task.

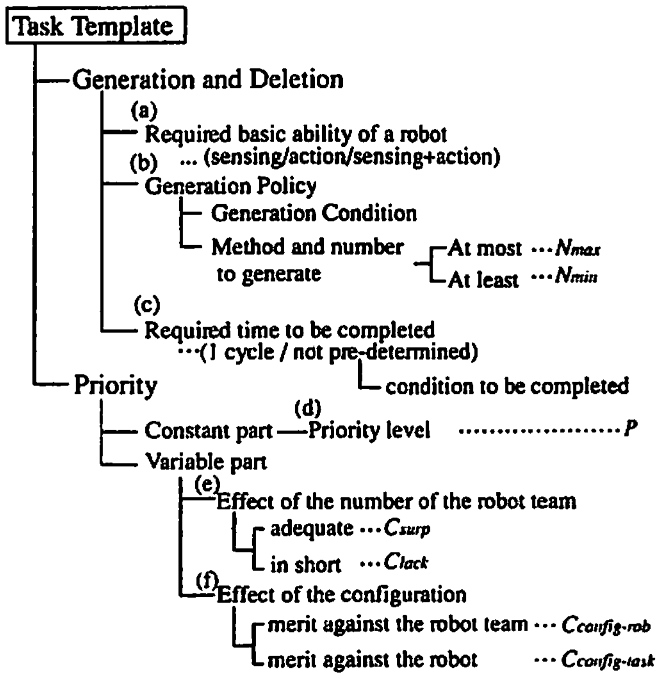
Task Template
Generation and Deletion
(a)
Required basic ability of a robot
(b) ... (sensing/action/sensing+action)
Generation Policy
Generation Condition
Method and number to generate
At most ...N_{max}
At least ...N_{min}
(c)
Required time to be completed
...(1 cycle / not pre-determined)
condition to be completed
Priority
(d)
Constant part — Priority level .... P
Variable part
(e) Effect of the number of the robot team
adequate ...C_{surp}
in short ...C_{lack}
(f) Effect of the configuration
merit against the robot team ... $C_{config\text{-}rob}$
merit against the robot ... $C_{config\text{-}task}$

Fig. 3. Task template.

Therefore, we propose an algorithm that relaxes the constraints from the lower priority level until it becomes solvable and optimizes the performance index PI (1). Here, we use the branch-and-bound method to solve the equation

$$PI = \sum_{i}^{n_{\mathrm{instance}}} \sum_{j}^{n_{\mathrm{freerob}}} w_{ij} X_{ij} \tag{1}$$

where

n_{instance}	total number of task instances;
n_{freerob}	the number of free robots;
X_{ij}	S_{ij} or A_{ij}.

The weight coefficient w_{ij} refers to the evaluation value of the status in which the robot j does the task instance i. Based on the dynamic part of the priority defined in the corresponding template, it is calculated as follows:

$$w_{ij} = C_0 \cdot C_{\mathrm{num}\cdot i} \cdot C_{\mathrm{config-task}\cdot i} + C_1 \cdot C_{\mathrm{config-rob}\cdot ij}$$
$$C_{\mathrm{num}} = \begin{cases} C_{\mathrm{surp}}, & \text{when } n_{\mathrm{freerob}} > N_{\mathrm{min}} \\ C_{\mathrm{lack}}, & \text{otherwise} \end{cases}$$

where C_0, C_1 are the weight coefficients ($C_0 = 100.0$, $C_1 = 1.0$ in the simulation of this paper), and N_{min} is the minimum number of task instances to be generated [Fig. 3(b)].

Because of the effect of the constraint equation, a task instance with a high priority level will be assigned to a robot without fault. On the other hand, the assignment system can still try to assign as many robots as possible to a task in a lower priority level by the effect of the performance index if C_{lack} or $C_{\mathrm{config-task}\cdot ij}$ has a high value. Therefore, it is possible to cope with a case in which there are too many task instances against the number of available robots.

TABLE I
TASK TEMPLATES FOR IMPLEMENTED TASKS

	Search Around	Displace Obstacle	Handling
(a) Basic ability	sensing	sensing + action	action
Condition to generate	When the information is older than T_{old}	When the removable object is found inside the circle to consider	Until the handling object reaches the goal configuration
(b) N_{max}	The following + $n_{freerob}$	The number of the removable obstacle	$n_{freerob}$
N_{min}	The number of the group of cells which meet the above condition	The number of the removable obstacle	2
term	1 cycle	not pre-determined	1 cycle
(c) Condition to terminate	—	When the obstacle is removed out of the handling path	—
(d) P	1	2	3
(e) C_{surp}	1.0	1.0	4.0
C_{lack}	8.0	8.0	1.0
(f) $C_{config\text{-}rob}$	$\alpha\ \times$(distance between robot i and the center of the area that corresponds to the task-instance j)	$\alpha\ \times$(distance between robot i and the center of the obstacle that corresponds to the task-instance j)	$\alpha\ \times$(distance between robot i and the center of the position to grasp that corresponds to the task-instance j)
$C_{config\text{-}task}$	Average timer count	1.0	1.0

where $\alpha = \begin{cases} 1: \text{no occlusion} \\ 10: \text{with occlusion} \end{cases}$

IV. IMPLEMENTATION OF TASK

In this section, we explain three kinds of tasks, "search around," "displace movable object," and "handle an object." These are implemented for the transport simulation in Section V and for the transport experiment in Section VI. Table I shows the contents of the task templates for these three tasks.

On implementing these tasks, we assume the following:

- robots can detect the attributes of a nearby obstacle (distance to obstacles, obstacles' shape) inside the limited stretch of their sensing area;
- robots can judge whether the detected object is removable or not based on the attributes.

First, we explain how to define the priority level, which expresses the relationship of the execution order [Table I(d)]. In transport work, a "handle an object" task most directly contributes to achieve the goal configuration. However, in order to "handle an object" safely without colliding with obstacles, it is required to "displace movable objects" first. This requires movable objects to be "searched" before being displaced. Therefore, the priority level was determined as shown in Table I.

Below is the explanation of each task. As the example of how to define other factors of task templates, we express the process to settle the parameters for the "displace movable object" task in detail in Section IV-B. A similar process is followed for other tasks.

A. "Search Around" Task

If the working environment changes, newer information about a certain area is credible and preferable. To generate task instances, it is necessary to derive the area to be searched. The area should be divided into smaller parts that can be searched by one robot. For this purpose, a work area is divided into small cells and a timer is set to each cell. The timer counts the elapsed time since the area was last searched to express how old the information is. From the limited area around the transport object, we take out cells whose timer shows time exceeding a threshold [gray cells in Fig. 4(c)]. The cells are divided into several groups that can be searched by one robot in one cycle and one task instance will be generated for each group. When deriving the above groups of cells, we divide the limited area around the transport object in a radial pattern and generate the histogram by counting the number of the included gray cells [Fig. 4(a)]. To treat adjacent cells as the same group, the histogram is converted into a sequence of 1 or 0, and the contraction and extraction operation is executed [Fig. 4(b)].

B. "Displace Movable Objects" Task

We assume that the object is small enough to be displaced by one robot. The detected removable object will be treated as the one which should be displaced if it is within the circle with a R_c radius around the center of the handling object.

The span of execution until the instance is deleted is set to "indefinite." This is because the robot that is once assigned this task goes on displacing the same object until the robot declares the completion of its task.

Here, we show how to set the dynamic part of the priority. As for the factor related to the number of robots, $C_{surp} < C_{lack}$ because it is necessary to assign as many robots as possible to clear the path even if the number of robots is smaller than the

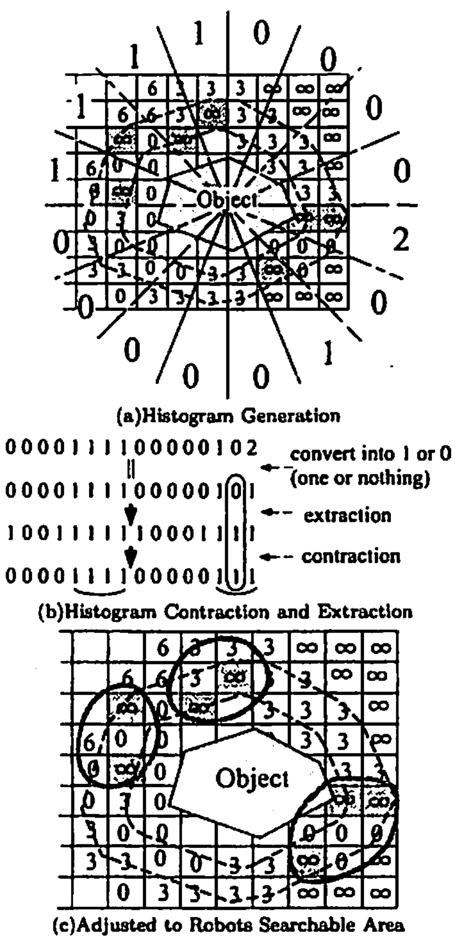
(a)Histogram Generation
convert into 1 or 0 (one or nothing)
extraction
contraction
(b)Histogram Contraction and Extraction
Object
(c)Adjusted to Robots Searchable Area

Fig. 4. How to generate task instances in the "search around" task. (a) Histogram generation. (b) Histogram contraction and extraction. (c) Adjusted to robots' searchable area.

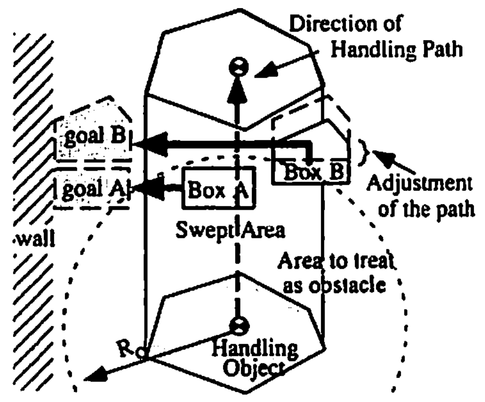

Fig. 5. Displacing path in the "displace movable objects" task.

number of removable objects. The actual value is determined by trial and error based on this guide of parameter design (the same parameters in other tasks are also determined similarly by trial and error). With regard to the factor related to the time and spatial configuration ($C_{\text{config-rob}}$, $C_{\text{config-task}}$), the parameters are determined as shown because of the following reasons: it is preferable to assign the nearest robot to save time to clear the path, and it is necessary to clear the vicinity of the transport object first before continuing manipulating it. To mainly focus on the task assignment method in this paper, we use a very simple strategy for paths to displace objects. A robot first approaches a removable object, and then displaces it vertically to the path of the handling object. The robot continues trying to displace the object until the object is no longer in the area that can be swept by the motion of the handling object. The other robots manipulate the handling object from the current configuration to the goal (or subgoal) without changing its orientation. If there are several removable obstacles, the removing path is adjusted so that the final destinations do not overlap with each other (Fig. 5).

C. "Handle an Object" Task

Here, we assume that an object is put on a truck and that robots handle the object by pushing or pulling it, generating adequate strength of two-dimensional force in a horizontal plane. We regard handling the object at each grasping position within one cycle as a task instance.

As for the motion planner of this task, a handling object path is planned as follows, given the prior goal configuration. Before the start, the goal configuration is added to the subgoal list.

Step 1) Update target subgoal if needed by checking whether the handling object reaches the current target subgoal.

Step 2) Make the path that connects the current configuration and the target subgoal.

Step 3) If unremovable objects lie in the swept area that is generated by moving the object from the current configuration to the next target subgoal in a beeline, adjust the path by adding a subgoal to avoid it in the direction vertical to this.

Obeying the above algorithm, a direction to manipulate is determined. Robots handle the object at constant speed, in principle, along this direction until the object reaches the next subgoal configuration. In the case that the following conditions are satisfied, the mechanism that suppresses moving works and the object is expected to stay still:

1) when some of the robots are executing the "displace movable objects" task on the path;
2) if any part of the area that should be searched remains unsearched in a "search around" task executed in the previous cycle.

V. Cooperative Transport Simulation

A. Contents of Simulations

In order to show the efficiency of the proposed method, cooperative transport simulations were carried out in the environment with unknown obstacles. Simulations are made for two robot groups that differ only in the total number of robots: group A has four robots and group B has two robots. A simulation for robot group A in another configuration is also made. Other simulation conditions as well as the environment are the same (Table II). Most of these conditions, including the sample rate, are given here so that they reflect those in the experiment in Section VI. Robots estimate whether the observed obstacle is removable or not by its size.

The proposed architecture can be implemented in the following two styles from the viewpoint of "who plans:"

TABLE II
SIMULATION CONDITIONS

the shape of the object	a 1.2 × 0.6-[m] rectangle
the shape of the movable object	a 0.6(w)×0.4(d)-[m] box
the shape of the robots	circle with a 0.2-[m] radius
sensing area of the robots	a sector with a 1.5-[m] radius and a 20-degree interior angle
maximum speed (handling)	0.1 [m/s]
maximum speed (running alone)	0.2 [m/s]
division of work space for sensing	a 0.2×0.2-[m] rectangle
sampling time	3.0 [s]

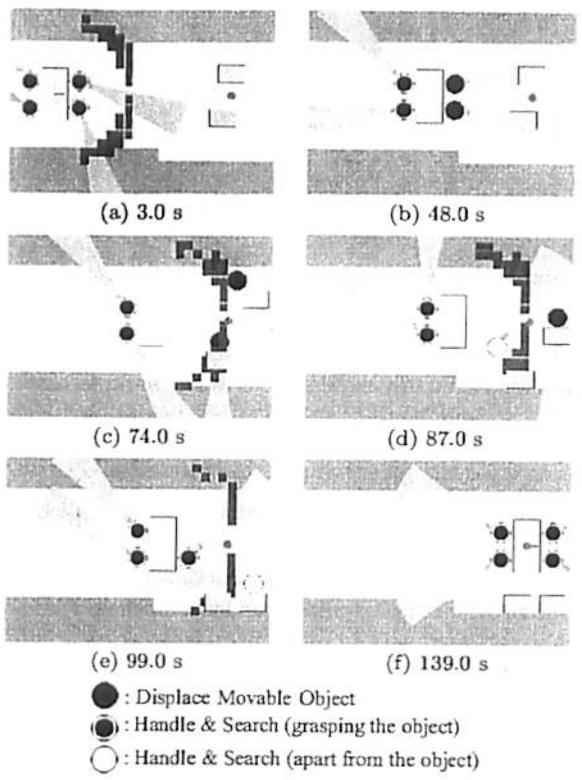
(a) 3.0 s
(b) 48.0 s
(c) 74.0 s
(d) 87.0 s
(e) 99.0 s
(f) 139.0 s
● : Displace Movable Object
● : Handle & Search (grasping the object)
○ : Handle & Search (apart from the object)

Fig. 6. Results of Simulation 1–four robots.

- one of the robots collects and integrates the other's environmental information. It plans the motion of the whole group and then communicates the results to other robots;
- all the robots have the same planning architecture. They exchange and integrate environmental information among themselves. Each robot plans the motion of the whole group and executes the corresponding part for itself.

Here, we implemented our system as the former style.

B. Simulation Results (Time Transition)

Simulation results are shown in Figs. 6–8. In each picture, a sector of each robot is its sensing area. A group of small squares in gray color shows the area that needs to be searched, which is used in generating "search around" task instances (see Section IV-A for details). The smallest circle indicates the goal position of the object.

In Simulation 1, robot group A (with four robots) achieves the whole task as follows. At the beginning, they continue handling the object while searching around. After one of the robots finds the obstacle, two of the four robots leave the object and go on to the "displace movable objects" task at 48 s [Fig. 6(b)]. While these robots are removing the obstacles, the rest of them continue transferring [Fig. 6(c)-(d)]. They finish removing obstacles, and then go back to the object to "handle" again [Fig. 6(e)]. Finally, all robots rejoin and "handle an object" until they reach the goal [Fig. 6(f)]. In the case of robot group B (with two robots) in Simulation 2, the process is different. After they find obstacles, two of the robots (that is, all the robots in a group) leave the object and go on to the "displace movable objects" task [Fig. 7(b)-(c)] at 54 s. While two robots are removing obstacles, the object stays still. Even if one of the robots goes back

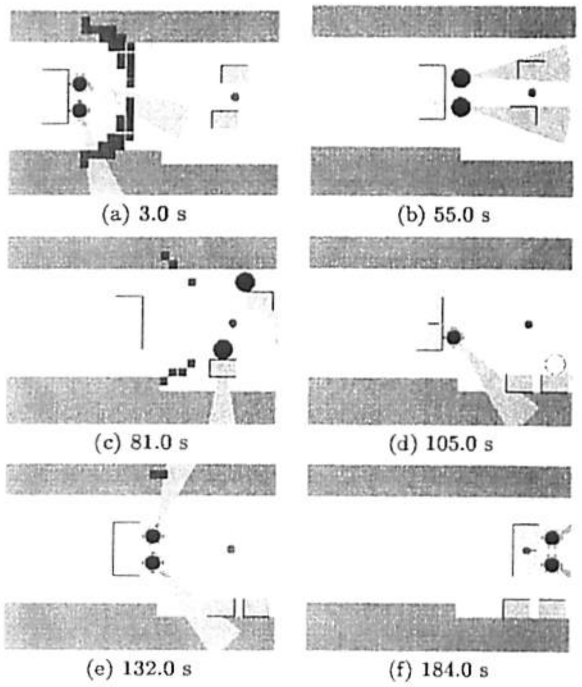

Fig. 7. Results of Simulation 2–two robots.

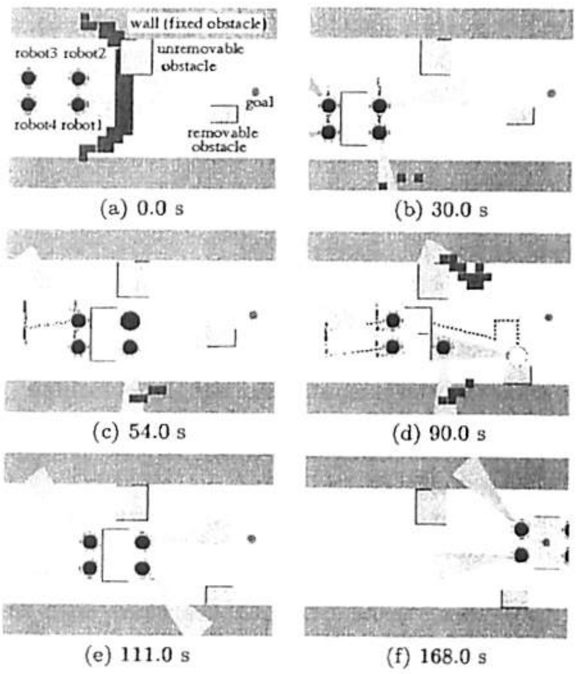

Fig. 8. Results of Simulation 3–four robots when an irremovable object exists.

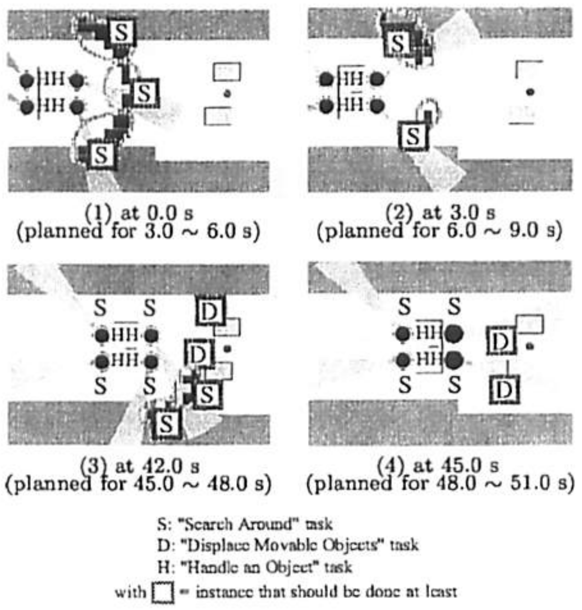
(1) at 0.0 s (planned for 3.0 ~ 6.0 s)
(2) at 3.0 s (planned for 6.0 ~ 9.0 s)
(3) at 42.0 s (planned for 45.0 ~ 48.0 s)
(4) at 45.0 s (planned for 48.0 ~ 51.0 s)
S: "Search Around" task
D: "Displace Movable Objects" task
H: "Handle an Object" task
with □ = instance that should be done at least

Fig. 9. Dynamically generated task instances.

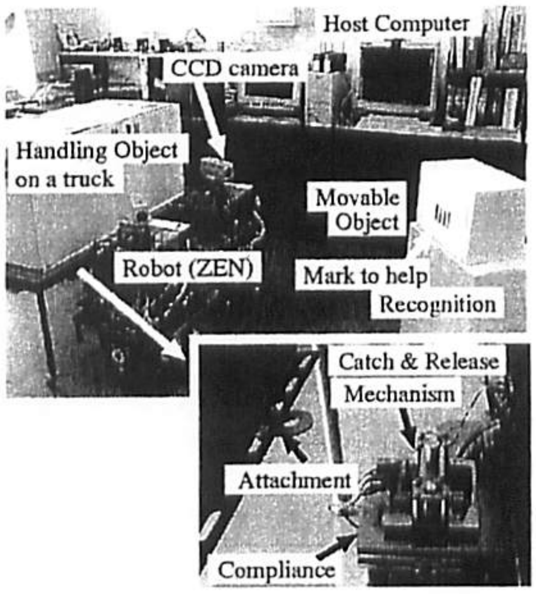

Fig. 10. Experimental setup.

to the object, it is impossible for only one robot to move the object. Therefore, the robot waits for another robot in order to move the object [Fig. 7(d)]. Finally, the other robot comes back to the object [Fig. 7(e)]. All the robots begin to "handle an object" again and reach the goal [Fig. 7(f)].

Fig. 8 shows the result of Simulation 3: The transport by robot group A under the existence of an unremovable object. At 3 s, robot 2 finds the unremovable object, and robots must change the transport path. The transport path being changed to avoid collision can be observed in Fig. 8(b).

Fig. 9 shows the dynamically generated task instances during the execution in Simulation 1.

From these simulation results, the following is shown:

- though the total execution time is different in Simulation 1 and Simulation 2 (by robot groups A and B, respectively), the robots' motion is planned to achieve the goal configuration in both cases;
- as for the one-step calculation time, it took about 0.11 s in Simulation 1 (four robots) and about 0.07 s in Simulation 2 (two robots) using SPARCstation20;
- Fig. 9 shows that various task instances which differ in kind and quality are generated as their work progresses;
- in Simulation 2, robot group B includes an inadequate number of robots for the required number of tasks, and we can see the strong effect of the factor of the number of robots. As shown in Fig. 7, robots abandon the task they are undertaking (searching and handling) and do another task (removing an obstacle).
- in Simulation 1, we can see the strong effect of the factor of the surroundings more often than that of the factor of the number of robots. As shown in Fig. 7(c)-(d), though they can afford to be assigned task instances to remove an object, they are first assigned the task instances to search around;
- as a result of the mechanism to suppress moving, the following series of actions was often observed in both groups: moving to search for the obstacles; stopping to observe the area that remains unsearched; and starting to move again [e.g., Fig. 9 from (1) to (2)]. This means that the area that needs to be searched does not continue increasing by a failure in the search task.

These show that the proposed architecture can be applied to an unknown environment where the number of task instances often exceeds the number of available robots.

In the simulation, the spillover of task instances was solved as time passed in the appropriate order of execution according to the number of available robots. This is due to the proposed dynamic priority-based approach which functions as the following three steps: 1) the generation of the task instances can be controlled by suppressing the object motion, the center of the robot group; 2) a task instance with higher priority will be selected and assigned earlier; and 3) even if the instance has low priority at the beginning, the priority will become higher as time advances if the instance remains important. This approach does not function appropriately in the case that 1) does not stand and the number of the task instances increases faster than the robots' execution, which is likely to occur if extended to the dynamic environment. In that case, additional mechanism to suppress the generation of task instances is necessary.

VI. Experiment

A cooperative transport experiment is carried out to show that the proposed method works effectively for the real robot system with which the error in positioning or measuring and the saturation of communication are associated. The experimental setup includes two real robots and a removable object whose position is unknown beforehand.

A. Experimental System

Fig. 10 shows the experimental system. Task-assignment planner and motion planners are implemented on a host computer (SPARCstation20) and the robots are connected with it

TABLE III
EXPERIMENTAL CONDITIONS

the shape of the object	a 1.2×0.6-[m] rectangle
the shape of the movable object	a 0.6(w)×0.4(d)×0.8(h)-[m] box
the size of the robots	0.4(w)×0.5(d)×0.7(h)-[m]
sensing area of the robots	a sector with a 2.0-[m] radius and a 20-[degree] interior angle
maximum velocity of the robots	0.1 [m/s^2]
division of work space for sensing	a 0.2×0.2-[m] rectangle
sampling time of the host computer	3.0 [s]

by wired ethernet LAN. Each robot controls itself on the basis of the received command (task and target position) and then reports the environmental information it has acquired.

We used two omnidirectional mobile robots, ZEN, developed by RIKEN. An object to transport is put on a truck and the robots handle the truck by pushing or pulling. Therefore, at least two robots are needed for handling.

Based on the authors' previous work [15], we introduce a compliance unit for the robot's hand that enables the robot to avoid generating too much internal force on the object as a result of positioning error. A stroke of the compliance unit is ±25 [mm]. In addition, to change the grasping state easily, a stick-type slider that moves up and down is put on this compliance unit. Each robot inserts its stick slider into one of the holes of the object to "grasp" and takes it out of there to "release" (Fig. 10).

As for environmental recognition, we put a mark on the side of a movable object and give robots its size and the meaning (the corresponding model of the object) beforehand. Each robot can, therefore, acquire the relative position and attributes of the object, respectively, and can estimate whether it can be removed or not by measuring the mark with its charge-coupled device (CCD) camera.

Experimental conditions are shown in Table III. The minimum radius of the sensing area is determined on the basis of the size of the mark. The maximum radius is determined so as to keep the success rate over 50%. The sample rate is determined based on the necessary time in the environmental recognition process using the CCD camera: the time to turn its head to the target position; and the time to capture the image and to derive the position information from it.

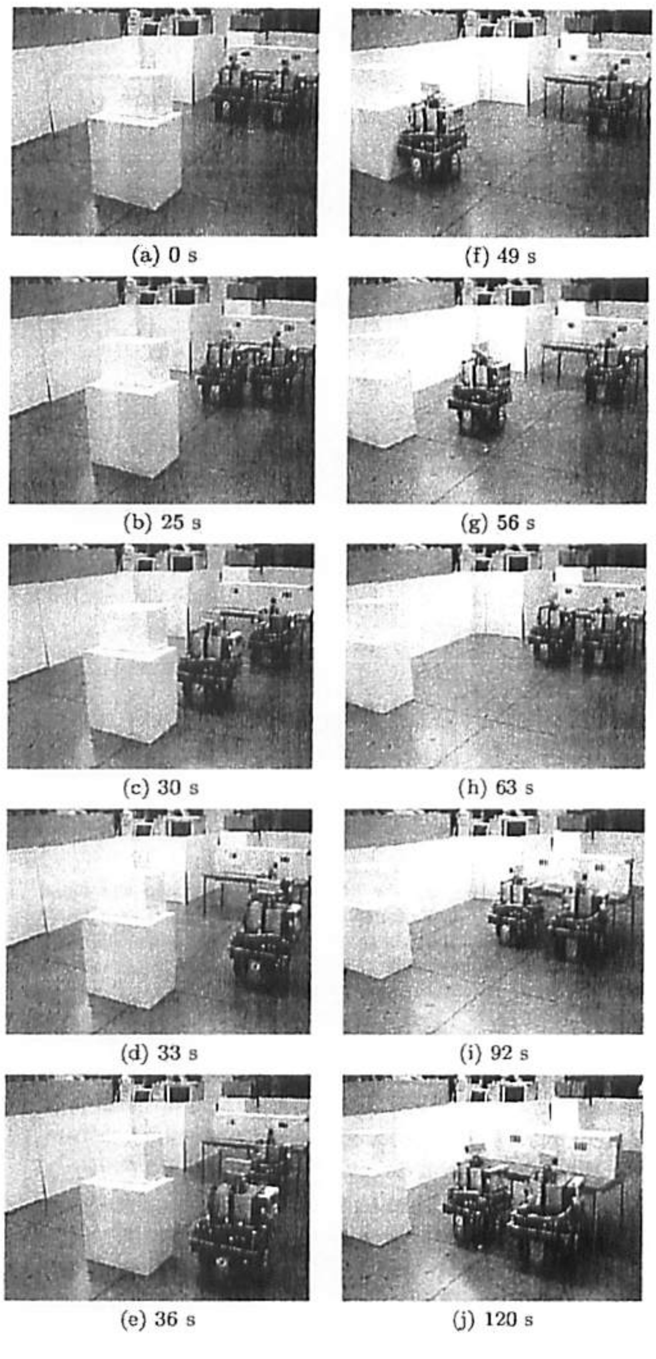

Fig. 11. Experimental results.

B. Experimental Results and Discussion

Cooperative transport was realized in an unknown static environment by a real robot system on which the proposed architecture is implemented.

Fig. 11 includes snapshots of the experiment. Fig. 12 shows the configuration of the robots, the handling object, and the movable obstacle. Generated and assigned task instances at each cycle are shown in Fig. 13.

Robots start to transport after being given the map in Fig. 12(a) beforehand. Both of the robots "searched around" [Fig. 11(a)]. At 3 s, robot 1 found the movable object [Fig. 11(b)]. At this time, the detected object was not included in the area to be treated as an obstacle. From 12 s, the robots started and continued moving (that is, handling) for 3 s. Then, the detected object came to be regarded as an obstructing object, and the "displace movable object" task instance was generated [Fig. 12(c)]. After finishing the search of the urgent area, robot 2 left the object and went on to the "displace movable objects" task. While robot 2 was displacing the obstacle [Fig. 11(c)–(g)], robot 1 stayed still just "searching around" and waited for robot 2 to handle the object together. At 51 s, robot 2 finished "displacing movable object" and declared that fact [Fig. 12(d)]. At 54 s, robot 2 was assigned to "search around," and went back to the handling object to "handle" again. Robot 1 corrected its positioning error because of the dead reckoning by measuring the landmark put on the handling object [Fig. 12(e)], and grasped the object at 72 s [Fig. 11(h)].

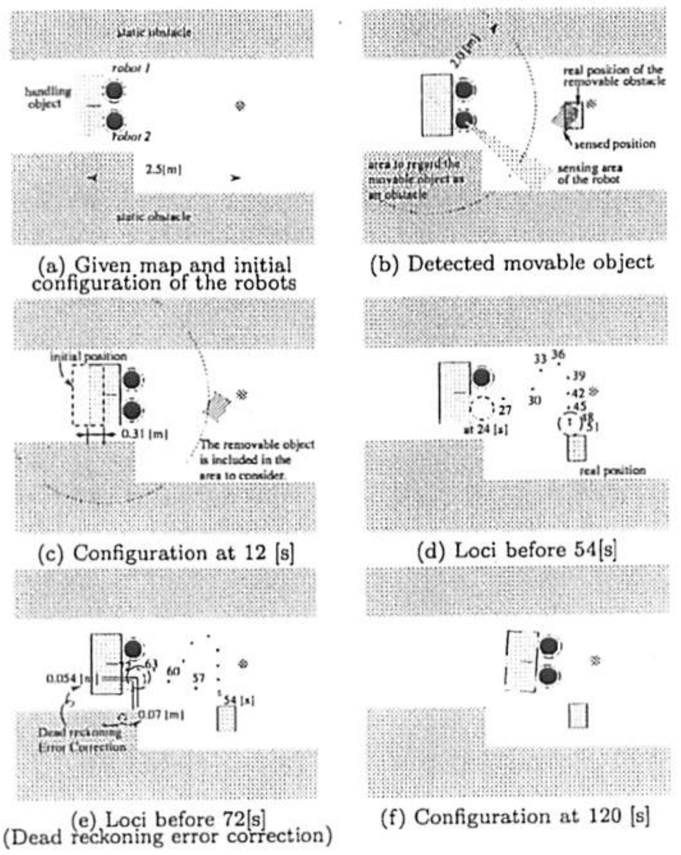
static obstacle
robot 1
handling object
robot 2
2.5[m]
static obstacle
(a) Given map and initial configuration of the robots
2.0[m]
real position of the removable obstacle
sensed position
area to regard the movable object as an obstacle
sensing area of the robot
(b) Detected movable object
initial position
0.31 [m]
The removable object is included in the area to consider
(c) Configuration at 12 [s]
at 24 [s]
real position
(d) Loci before 54[s]
0.054 [m]
0.07 [m]
54 [s]
Dead reckoning Error Correction
(e) Loci before 72[s] (Dead reckoning error correction)
(f) Configuration at 120 [s]

Fig. 12. Loci of the robots and the objects. (a) Given map and initial configuration of the robots. (b) Detected movable object. (c) Configuration at 12 s. (d) Loci before 54 s. (e) Loci before 72 s (dead reckoning error correction). (f) Configuration at 120 s.

Both of the robots went on handling until a total of 120 s passed from the beginning [Fig. 11(i), (j)]. Because of the lack of feedback concerning the displacement of the compliance unit of each robot's hand, the displacement was about 17 mm from the center along the vertical direction to the handling path.

Though the duration of the experiment was not enough to complete the given goal configuration [Fig. 12(f)], the results show a basic process of transport associated with the task assignment.

As we mentioned at the beginning of this section, a real robot system involves error in the sensing process. From Fig. 13, we can see that the robots detected the same object twice. The measurement differed at each time. To prevent the robots from treating them as different objects, we set an area around the already detected object and checked the possibility of duplication in this experiment. This also suggests the robustness of our real-time replanning system against the failure in finding unknown objects at a glance. Even if a robot fails, there is a high possibility that it will find them in the next or another cycle. In addition, such an aspect can also solve the differences between measured and real configurations [e.g., in Fig. 12(b)] by revising them as the transport goes on.

Finally, two robots could communicate with each other effectively without saturation as they were planning motion and executing. It took about 0.03 s in calculation and also about 0.03 s in communication on average for each sampling cycle. These results show that the proposed architecture can be used in a real robot system.

time	the number of generated task instances				role of each robot		object motion	
[s]	Ⓢ	S	Ⓓ	Ⓗ	rob1	rob2	status	
0	—	—	—	—	—	—	—	Rob1 found the obstacle.
3	5	0	0	2	ⓈH	ⓈH	stop1	
6	4	0	0	2	ⓈH	ⓈH	stop1	
9	3	0	0	2	ⓈH	S H	stop1	
12	0	2	0	2	S H	S H	move	Rob1 found the obstacle again.
15	4	0	1	2	ⓈH	ⓈH	stop1	
18	5	0	1	2	ⓈH	ⓈH	stop1	
21	2	0	1	2	ⓈH	ⓈH	stop1	
24	0	2	1	2	S H	Ⓓ	stop2	Rob2 started displacing the obstacle.
27	1	0	0	1	ⓈH			
30	0	1	0	1	ⓈH			
33	0	1	0	1	S H			
36	0	1	0	1	S H			
39	1	0	0	1	ⓈH			
42	1	0	0	1	ⓈH			Rob2 finished displacing the obstacle.
45	0	1	0	1	S H			
48	0	1	0	1	S H			
51	3	0	0	1	ⓈH			
54	3	0	0	1	ⓈH	Ⓢh		
57	4	0	0	1	ⓈH	Ⓢh		Rob2 was returning to the object.
60	1	1	0	1	S H	Ⓢh		
63	1	1	0	1	ⓈH	Ⓢh		
66	0	2	0	1	S H	S h		
69	0	2	0	1	S H	S h		
72	0	2	0	1	S H	S h		Rob2 grasped the object again.
75	3	0	0	1	ⓈH	ⓈH	move	
78	5	0	0	1	ⓈH	ⓈH	stop1	
81	3	0	0	1	ⓈH	ⓈH	stop1	
84	0	2	0	1	S H	S H	move	
87	5	0	0	1	ⓈH	ⓈH	move	
90	5	0	0	1	ⓈH	ⓈH	stop1	
93	3	0	0	1	ⓈH	ⓈH	stop1	
96	3	0	0	2	S H	ⓈH	stop1	
99	1	1	0	2	S H	ⓈH	stop1	
102	0	2	0	2	S H	S H	move	
105	5	0	0	2	S H	ⓈH	move	
108	5	0	0	2	ⓈH	S H	stop1	
111	5	0	0	2	ⓈH	ⓈH	stop1	
114	3	0	0	2	ⓈH	S H	stop1	
117	1	1	0	2	ⓈH	ⓈH	stop1	

S: "Search Around" task
D: "Displace Movable Objects" task
H: "Handle an Object" task
└h : when the assigned robot is apart from the object
○: instance that should be done at least

Stop1 : stop to sense remaining unsearched area
Stop2 : stop to wait for another robot to come back

Fig. 13. Generated task instances and the assigned results.

Here, we discuss the relationship between the complexity of the problem and the calculation time. As mentioned in

Section III-C, the assignment algorithm is implemented using the branch-and-bound method in the simulations and the experiment in this paper. This kind of 0–1 integer programming is essentially NP-complete, however, the calculation time can be reduced to be polynomial using the approximate solution method such as pivot-and-complement method [24]. The calculation time is experimentally revealed to be approximately $O(n^{4.7})$ using a SPARC4/370, where n means the number of variables [25]. Roughly estimating, up to about six robots can be used in this experimental setup on the following assumptions: the approximate solution method is used to solve the problem; the calculation cost is proportional to the fourth power of the number of variables n (at worst $n \approx n_{\mathrm{rob}} \times n_{\mathrm{instance}}$ where n_{rob} means the number of all the robots in the group); the communication cost is proportional to the number of robots; and the sampling time is equal to 3 s, which was determined based on the hardware constraints.

Next, discussion is about the effect that our architecture allows the long-period type of task. In the present implementation of the task, once the robot is assigned a "displace movable objects" task instance, it goes on displacing until the object is moved out of the handling path. This works well to avoid deadlock and chattering between this task and, for example, a "handle an object" task. From another point of view, however, it means that our system is slow in the change of the assignment strategy according to the surroundings when the long-period type of task instance is assigned. To deal with this aspect, it is necessary to add the upper-level process that monitors the progress of the task execution to evaluate the strategy of the robots as a group.

Here, we show how to design the parameters, C_{surp} and C_{lack}, that define the task assignment strategy according to the number of the robots. First of all, several typical situations are supposed. Variation of the following elements should be examined: the number of the available robots; the disposition of the robots; and the object and other environmental conditions related to the surroundings. The designer should first determine the preferable assignment result for each situation. Then the designer calculates the values of the variables for each situation, so that the given preferable assignment is realized. To know the range of the variables, it is better to calculate several combinations of the variables for each situation. The final values are determined to satisfy all the situations derived above.

VII. Conclusion and Future Work

This paper proposed a task-assignment method for cooperative transport by multiple mobile robots in an unknown static environment.

- We dealt with the task to be assigned by dividing a robot's actions into small units so that one unit would be performed by one robot in a short period of time. By inputting sensor information to the "task templates," tasks needed at each cycle are dynamically generated as task instances.
- Each assignment candidate is evaluated from the following two points of view: whether the number of robots available for cooperation, and the time and spatial distribution of task instances and robots, are adequate.

This enables us to formulate a task assignment problem at each cycle that we call an "assignment problem." Then, the problem can be solved in real time.

This task assignment process will be repeated at a constant sampling rate. Each assignment process is associated with motion planning independently for each task. Thanks to the division into small periods of time, the motion-planning process only needs a simple estimation.

The results of the transport simulation and the experiment showed the validity of the proposed assignment algorithm.

Though our assignment method has been developed to deal with a static environment, it could be applied to a dynamic environment by extending the task definition: for example, by adding information to each motion planner on how to react when the moving object is found. Such an extension is one of the most essential issues in developing real robot system.

References

[1] H. Kimura, Z. Wang, and E. Nakano, "Huge object manipulation in space by vehicles," in *Proc. IEEE/RSJ Int. Workshop Intelligent Robots and Systems*, vol. 1, 1990, pp. 393–397.

[2] M. J. Mataric *et al.*, "Cooperative multirobot box-pushing," in *Proc. IEEE Int. Conf. Robots and Systems*, vol. 3, 1995, pp. 556–561.

[3] O. Khatib, "Mobile manipulation: The robotic assistant," *Robot. Auton. Syst.*, vol. 26, no. 2–3, pp. 175–183, 1999.

[4] H. Osumi, "Cooperative strategy for multiple position-controlled mobile robots," in *Distributed Autonomous Robotics Systems 2*. New York: Springer-Verlag, 1996, pp. 374–385.

[5] J. Desai, C.-C Wang, M. Zefran, and V. Kumar, "Motion planning for multiple mobile manipulators," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1996, pp. 2073–2078.

[6] D. J. Stilwell and J. S. Bay, "Toward the development of a material transport system using swarms of ant-like robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1993, pp. 766–771.

[7] ——, "Optimal control for cooperating mobile robots bearing a common load," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1994, pp. 58–63.

[8] M. Hashimoto, F. Oba, H. Nakahara, K. Imamaki, and T. Eguchi, "Trajectory generation and tracking control methods for a multiple transfer robots system," in *Proc. IEEE/RSJ Int. Workshop Intelligent Robots and Systems*, 1991, pp. 799–804.

[9] M. Hashimoto, F. Oba, and T. Eguchi, "Dynamic control approach for motion coordination of multiple wheeled mobile robots transporting a single object," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1993, pp. 1944–1951.

[10] K. Kosuge *et al.*, "Transportation of a single object by two decentralized-controlled nonholonomic mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1998, pp. 2989–2994.

[11] ——, "Distributed robot helpers handling a single object in cooperation with a human," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2000, pp. 458–463.

[12] Z.-D. Wang, E. Nakano, and T. Matsukawa, "Cooperating multiple behavior-based robots for object manipulation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1994, pp. 1524–1531.

[13] G. Ogasawara, T. Omata, and T. Sato, "Multiple movers using distributed, decision-theoretic control," in *Proc. Japan–USA Symp. Flexible Automation*, vol. 1, 1992, pp. 623–630.

[14] N. Miyata *et al.*, "Cooperative transport with regrasping of torque-limited mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1996, pp. 304–309.

[15] ——, "Cooperative transport system with regrasping car-like mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1997, pp. 1754–1761.

[16] F. R. Noreils, "Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment," *Int. J. Robot. Res.*, vol. 12, no. 1, pp. 79–98, 1993.

[17] P. Caloud *et al.*, "Indoor automation with many mobile robots," in *Proc. IEEE Int. Workshop Intelligent Robots and Systems*, 1990, pp. 67–72.

[18] H. Asama *et al.*, "Development of a task-assignment system using communication for multiple autonomous robots," *J. Robot. Mechatron.*, vol. 4, no. 2, pp. 122–127, 1992.

[19] T. Vidal *et al.*, "Incremental mission allocation to a large team of robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1996, pp. 1620–1625.

[20] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 220–240, Apr. 1997.

[21] T. Shibata *et al.*, "Spontaneous behavior of robots for cooperation—Emotionally inteligent robot system," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1996, pp. 2426–2431.

[22] Y. Kuniyoshi *et al.*, "Vision-based behaviors for multirobot cooperation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1994, pp. 925–932.

[23] D. Jung *et al.*, "An architecture for distributed cooperative planning in a behavior-based multi-robot system," *Robot. Auton. Syst.*, vol. 26, pp. 149–174, 1999.

[24] E. Balas and C. Martin, "Pivot and complement—A heuristic for 0-1 programming," *Manage. Sci.*, vol. 26, pp. 86–96, 1980.

[25] M. Ishizuka and T. Okamoto, "A polynomial-time hypothetical reasoning employing an approximate solution method of 0-1 integer programming for computing near-optimal solution," in *Proc. 10th Canadian Conf. Artificial Intelligence*, 1994, pp. 179–186.

Natsuki Miyata (M'00) received the B.E. degree in 1995, the M.E. degree in 1997, and the Ph.D. degree in 2000 in precision engineering from the University of Tokyo, Tokyo, Japan.

Her research interests include motion planning and the realization of cooperation among multiple mobile robots, as well as the modeling of individual behavior of the human.

Dr. Miyata is a member of RSJ and JSPE.

Jun Ota (M'92) received the M.S. and D.S. degrees from the Faculty of Engineering, University of Tokyo, Tokyo, Japan, in 1989 and 1994, respectively.

He is an Associate Professor in the Department of Precision Machinery Engineering, Graduate School of Engineering, University of Tokyo. From 1989 to 1991, he was with Nippon Steel Corporation, Kanagwa, Japan.. 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.

Tamio Arai (M'92) received the Ph.D. degree in engineering from the University of Tokyo, Tokyo, Japan, in 1977.

In 1987, he became a Professor in the Department of Precision Engineering, University of Tokyo. He was a Visiting Researcher in the Department of Artificial Intelligence from 1979 to 1981. His specialties are assembly and robotics, especially multiple mobile robots, including the legged robot league of RoboCup. He works on IMS programs in holonic manufacturing systems. He has contributed to the development of robot software in ISO activity. Since 2000, he has been a Director of Research into Artifacts, Center for Engineering, University of Tokyo.

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

Hajime Asama (M'90) received the M.S. and Ph.D. degrees from the Department of Precision Machinery Engineering, University of Tokyo, Tokyo, Japan, in 1984 and 1989, respectively.

He joined the Chemical Engineering Laboratory of RIKEN (the Institute of Physical and Chemical Research) in 1986, and is currently the Head of the Instrumentation Project Promotion Division and Extreme Conditions Mechatronics Team, and is a Senior Scientist in the Biochemical Systems Laboratory of RIKEN. He was a Visiting Associate Professor of Saitama University, Saitama, Japan, in 1999. His main interests are distributed autonomous robotic systems, cooperation of multiple autonomous mobile robots, emergent robotic systems, and intelligent data carrier systems. He served as an Editor of *Distributed Autonomous Robotics Systems* and its second volume which were published by Springer-Verlag, Tokyo, in 1994 and 1996, respectively.

Dr. Asama received the Promotion of Advanced Automation Technology Award in 1992, JSME Robomec Award in 1995, and the JSME Robomec Best Poster Award in 1996. 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, and the New York Academy of Science.