Random Sampling Algorithm for Multi-agent Cooperation Planning

Shotaro Kamio and Hitoshi Iba
Department of Frontier Informatics,
Graduate School of Frontier Sciences, The University of Tokyo,
5-1-5 Kashiwa-no-ha, Kashiwa-shi, Chiba, 277-8561, Japan.
{kamio, iba}@iba.k.u-tokyo.ac.jp

***Abstract*— The cooperation of several robots are needed for complex tasks. The cooperation methods for multiple robots generally require exact goal or sub-goal positions. However, it is difficult to direct the goal or sub-goal positions to multiple robots for the sake of cooperation with each other.**

Planning algorithms will reduce the burden for this purpose. In this paper, we propose a multi-agent planning algorithm based on a random sampling method. This method doesn't require the exact sub-goal positions nor the times at which cooperation occurs. The effectiveness of this approach is empirically shown by simulation results.

***Index Terms*— RRT, path planning, cooperation, multi-agent.**

I. Introduction

The cooperation of robots enables them to carry out complex tasks, such as transferring objects [1], efficiently. Hence, techniques in which multiple robots cooperate to perform tasks will be increasingly important in the future. In the direct robot operation, many instructions are needed to transfer the operator's intention to the robot which performs the target task. An appropriate use of planning algorithms can reduce the amount of these instructions. In addition, it realizes autonomous cooperation of robots.

In recent years, a technique called Rapidly-exploring Random Trees (RRT) has emerged as an algorithm useful for planning [2]. When the planning involves cooperative behavior among multiple robots, however, there are problems with the original RRT algorithm. In this paper, we propose a revised algorithm for planning the cooperative behavior among multiple robots. The target problem is transferring an object with the cooperation of robots.

This paper is structured as follows: following section describes the planning and the RRT technique. Section III explains the problem being targeted, and section IV outlines the proposed technique. Section V describes the experimental results of simulation to verify the proposed technique, section VI presents discussions about the results, followed by a conclusion in section VII.

II. Planning

Over the past few decades, RRT has been used in planning [2], [3]. This is a technique that allows planning to be carried out without pre-processing. This enables high-speed algorithms, and a large volume of research has been conducted in this area in recent years. The RRT algorithm will be explained in the next subsection.

RRT is used in the motion planning [4], [5] and path planning [6], [7] of the robot. Kuffner et al. used RRT to perform motion planning (grasping objects, lifting one leg, etc.) in a humanoid robot [4]. If the initial state, final state, environment model, and the robot model are provided, it can plan a series of states from the initial state to the final state without bumping into any surrounding objects. Okada et al. used the RRT for planning the path of a humanoid robot which avoided obstacles using stereo vision [7].

At present, there has been some research on using multiple robots in cooperation to carry items (for example, as described in [1]). However, in these research studies, heuristic planning algorithms were used or the goal position was provided in advance. Using our proposed method (section IV), we succeeded in generating sub-goals for the cooperation task automatically from small amounts of information.

If the task is more complicated, an operator will be forced to send more instructions to the robot in order for the task to be achieved. Within a similar framework, performing cooperative tasks increases the instructions required to operate the multiple robots because the number of robots increases. In order to put this into practice, ensuring the instructions provided to the robots are simple will be of benefit to the operator. Our proposed technique is useful in this regard as well.

A. Rapidly-exploring Random Trees (RRT)

RRT is a technique in which random sampling is used to grow search trees within a state space [2], [3]. Planning is based on these search trees.

1) Algorithm: The RRT algorithm builds a search tree τ that has the initial state x_{init} as its root, based on the following procedure [2]. First, a random state (x_{rand}) is chosen from the state spaces to be searched. Next, branches are extended from the search tree τ towards x_{rand}. This is either repeated a given number of times, or is repeated until the search tree arrives at the final state.

Because RRT offers almost no hypotheses in terms of the problem, we can easily applied it to various problems.

2) Extended techniques: Kuffner et al. have proposed the RRT-Connect [3], a technique for achieving bidirectional

```
RRT_CON_CON($x_{init}$, $x_{goal}$)
  for $k$ = 1 to K do
    $x_{rand} \leftarrow$ RANDOM_STATE();
    if not (CONNECT($\tau_a$, $x_{rand}$) = Trapped) then
      if (CONNECT($\tau_b$, $x_{new}$) = Reached) then
        return Path($\tau_a$, $\tau_b$);
    SWAP($\tau_a$, $\tau_b$);
  return Failure
```

Fig. 1. RRT-ConCon: an algorithm that carries out bidirectional searches with RRT.

growth of the search tree from both the initial state and the final state. Bidirectional searching is a technique that is often also used in typical search algorithms, and is extremely effective in improving the searching efficiency. Additionally, they have proposed a CONNECT operation in which the EXTEND operation is repeated only as long as the search tree is growing.

According to LaValle et al. [2], the CONNECT operation demonstrates the best search performance with holonomic constraints, and the EXTEND operation demonstrates the best search performance with non-holonomic constraints. When a bidirectional search is carried out, these operations can be combined, but of these, the best performance with holonomic constraints is demonstrated by the RRT-ConCon algorithm (Fig. 1) [2].

The planning algorithm proposed in section IV was implemented in the MSL (Motion Strategy Library) [8].

III. The Targeted Problem

The problem targeted in our research is the task of having humanoid robots cooperate in carrying a load from the start position to the goal position. In our project [9], two robots would cooperate and transfer the load from one to the other. The state of the environment at a certain point in the future should be known in advance.

In this study, we used three robots, each 48.3 cm tall with an arm length of 10 cm which is almost same as that of the humanoid robot HOAP-1 manufactured by Fujitsu Automation Limited. The states of the robots were expressed using 3D information (x coordinate, y coordinate, and rotation angle around z axis).

First, the robot which has the load at the initial position delivers it to another robot. The robot that receives the load then transfers it on to yet another robot. The last robot carries the load to the goal point. In order to transfer the load, the robots have to cooperate and move to positions that are close enough that the load can be transferred from one to another.

In this research, the following information was known in advance.

- Models of the environment and robots
- Goal position for the load
- Procedure for cooperation

In order to carry out the experiment, which is described later, we randomly created three environments: Environment

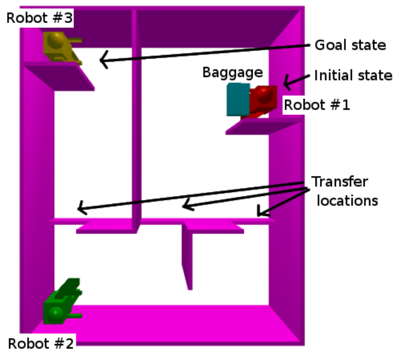


Fig. 2. Environment A

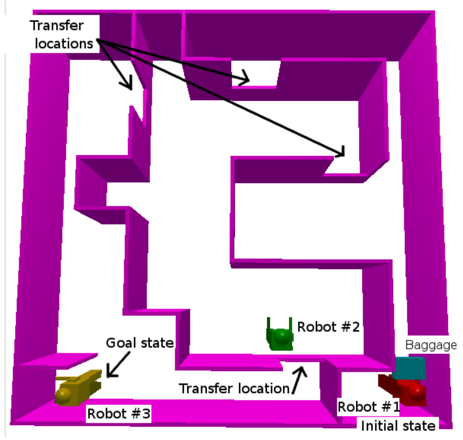


Fig. 3. Environment B

A (Fig. 2, 120 cm $\times$ 100 cm), Environment B (Fig. 3, 160 cm $\times$ 160 cm) and Environment C (Fig. 4, 160 cm $\times$ 160 cm). These environments are different in regard to symmetry property and complexity. Each robot was in separate rooms, divided by walls. The robots are not able to go beyond the walls, but they are able to transfer the load back and forth at points where the walls are low (points labeled “transfer location” in the figures).

The goal position for the load was provided, but not the goal positions for any of the robots. The cooperation procedure consisted of a series of sub-tasks in which the robots would cooperate, with the procedure indicated in Table I.

As conditions for a robot to pass the load to another robot, the two robots would reach to within a distance of 25 cm, and

TABLE I
Cooperation procedure

Sub-task no.	Content of sub-task
1	Load transferred from Robot 1 to Robot 2
2	Load transferred from Robot 2 to Robot 3
3	Load placed at goal position by Robot 3

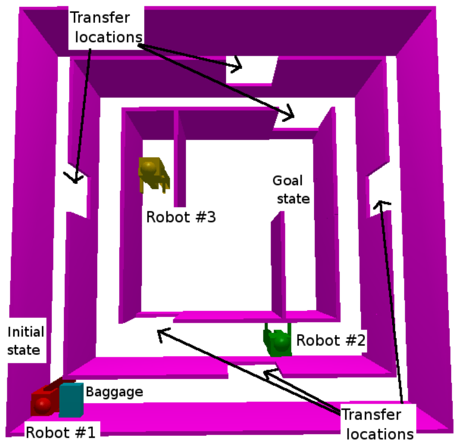


Fig. 4. Environment C

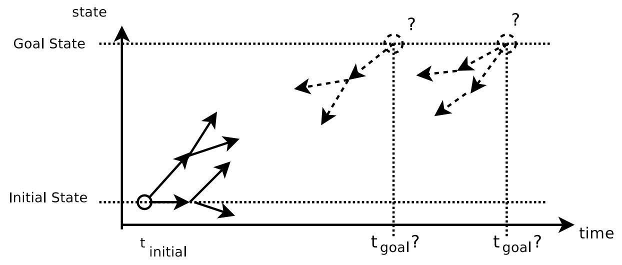


Fig. 5. When conducting a search that includes a time parameter, a bidirectional search is not possible unless the time to reach the goal state is decided (example shows a search in a one-dimensional state space).

```
plan_subtask($i$)
    if no_more_subtask($i$) then
        return Success;
    $r_1, r_2$ := robots_to_cooperate(i);
    for $j$ = 1 to MAX_RETRY do
        plan_ith_subgoal($i$, $r_1$, $r_2$);
        plan_subplan_for_robot($r_1$);
        plan_subplan_for_robot($r_2$);
        adjust_subplan_length($r_1$, $r_2$);
        plan_subtask($i + 1$);
        if status = Success then
            return Success;
    return Give_up;
```

Fig. 6. Algorithm that generates a sub-plan for the i-th sub-task. If errors occur in the routines within the "for" loop, it backtracks and retries the "for" loop, but this has been omitted here for purposes of simplification.

the offset from the angle at which they faced each other would be 0.2 radians (approximately 10 degrees) or less. These were the goal conditions for each sub-task and are hereafter referred to as the "sub-goal conditions".

Because the experiment targets humanoid robots, it is assumed that they could move freely within the surrounding area, and the environment is treated with the holonomic constraint.

The planning algorithm proposed in this paper is implemented in the MSL (Motion Strategy Library) [8].

A. *Problems that occur when the normal RRT is applied*

Simply using the normal RRT technique, a state space can be set that expresses three robots at one time for application to these environments. Then, the overall state space is 3 (robots) $\times$ 3 (dimensions per robot) $=$ 9-dimensional.

This situation, however, presents the following problems.

- The goal states for each of the robots need to be explicitly provided.
- Wasteful movements occur that are unrelated to the task.

The first problem is a necessary condition in order to conduct a bidirectional search. It is possible to provide only the condition that will serve as the goal state and then search until the goal state is reached, without conducting a bidirectional search. However, this requires an extremely long execution time, and is thus unrealistic.

The second problem occurs even if the goal states are explicitly provided for each of the robots. With straightforward RRT, the planning for all three robots is executed simultaneously. As such, even if one robot reaches a sub-goal for cooperation, it takes time for the other robot involved in the cooperation task to reach the sub-goal state, so the robot exhibits meaningless operations such as moving away from the state or moving closer to it. This is a problem that cannot be avoided unless planning is performed individually for each robot.

To reduce wasteful movements, searching needs to be performed separately for each individual robot. To realize this, a time parameter can be included as a state for each robot and searching carried out. When searching is carried out including a time parameter, however, it becomes difficult to conduct a bidirectional search, because, in order to have the search tree grow from the goal state side, the time to arrive at the goal needs to be defined, as indicated in Fig. 5. Generally, it is difficult to define in advance the time it will take to reach the goal. If a bidirectional search is not carried out, however, the searching efficiency drops sharply. We used the technique proposed in the following section to solve this problem.

IV. Proposed Technique

Our search technique is characterized by three features:

- There is no need to determine a final time or goal state for each of the robots.
- RRT can be used to automatically generate the sub-goals needed for the sub-tasks.
- In the planning for the sub-plans up to the sub-goals, a search tree in which time is considered, and one in which it is not, were used simultaneously. This enabled bidirectional RRT searching and improved the searching efficiency.

Here, the path plan up to the sub-goal for each robot working in cooperation is called the "sub-plan".

The planning algorithm is outlined by the following recursive procedures (Fig. 6):

```
plan_ith_subgoal($i$, $r_1$, $r_2$)
  for k = 1 to K do
    $s_{rand} \leftarrow$ RANDOM_STATE($r_1$);
    if not (CONNECT($\tau_1$, $s_{rand}$) = Trapped) then
      $s_{coop} \leftarrow$ cooperative_state_of($s_1$);
      CONNECT($\tau_2$, $s_{coop}$);
      foreach $s_2$ in $\tau_2$
        if acceptable_for_subgoal($s_1$, $s_2$) then
          return ($s_1$,$s_2$);
    $s_{rand} \leftarrow$ RANDOM_STATE($r_2$);
    if not (CONNECT($\tau_2$, $s_{rand}$) = Trapped) then
      $s_{coop} \leftarrow$ cooperative_state_of($s_2$);
      CONNECT($\tau_1$, $s_{coop}$);
      foreach $s_1$ in $\tau_1$
        if acceptable_for_subgoal($s_1$, $s_2$) then
          return ($s_1$,$s_2$);
  return Failure;
```

Fig. 7. Algorithm that generates the i-th sub-goal.

1) A pair of sub-goals corresponding to the i-th ($1 \leq i \leq N$) sub-task is generated for the two cooperating robots.
2) The sub-plan is generated for each of the two robots to reach their sub-goals.
3) The final times of the two sub-plans are aligned.
4) The procedure is executed recursively with respect to the $(i + 1)$-th sub-task.

An algorithm based on RRT is used to generate the sub-plans. Because RRT is a probabilistic algorithm, errors sometimes occur in the generation of sub-plans (assuming a sub-plan can even be generated). If an error occurs in generating a sub-plan, backtracking is carried out, and the process is redone from the sub-goal generation step. When the $(i - 1)$-th sub-goal was not generated properly, errors would occur many times in the generation of either the i-th sub-goal or the sub-plan. In that case, backtracking of the $(i-1)$-th sub-goal is carried out and the procedure is redone from the generation of the $(i - 1)$-th sub-goal.

A. Generating sub-goals

In generating sub-goals, planning is carried out in an environment in which robots other than the two cooperating robots do not exist. This is equivalent to giving priority in sub-plans to the two cooperating robots over other robots.

The algorithm for generating sub-goals is as shown in Fig. 7. The basic algorithm is the same as the RRT with bidirectional searching (RRT-ConCon). The respective RRTs are generated from the current states of the two robots that are to work cooperatively together. However, the bidirectional trees extend search trees towards each other, in the direction of a "state S_{coop} that satisfies the sub-goal conditions" (a distance of 20 cm between the two robots, with an offset of zero from the angle at which they face each other). If a pair of states that satisfy the conditions is found, those states are returned.

In this sub-goal generation, planning is performed without including a time parameter. Because no time parameter is included, high-speed planning is possible.

```
plan_subplan_for_robot($r$)
  repeat
    $t \leftarrow t + \delta t$;
    if is_collision_free($t$, $s_r$) and $s_r$ = $s_{subgoal}$ then
      add_state_to_plan($r$, $t$, $s_r$);
    else
      status := plan_toward_subgoal($r$, $t$, $s_r$);
      if not (status = Success) then
        return status;
  until $t + \delta t \leq t_{end}$;
```

Fig. 8. Algorithm that generates the sub-plan (r is the robot ID, t is the start time, s_r is the state at the start time, $s_{subgoal}$ is the sub-goal state of the robot r, and t_{end} is the end time of the sub-plan).

B. Generating sub-plans

After a sub-goal has been generated, the paths (sub-plans) taken by the respective robots to reach the sub-goal are created. In doing this, times are taken into consideration, and sub-planning is carried out in such a way that even if the other robots are moving at the same time, the robots do not collide.

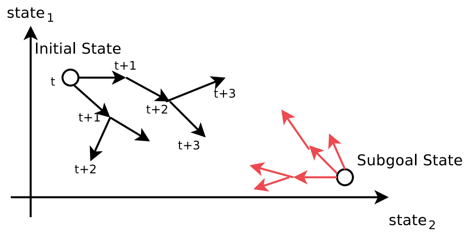
As shown by the "plan_subtask()" routine (Fig. 6) of the two robots (here called r_1 and r_2) that are working cooperatively together, planning is initially carried out to move the robot r_1 that has the load to the sub-goal. Planning is then carried out to move the other cooperating robot r_2 to its sub-goal without colliding with robot r_1. The planning for robot r_2 is done in such a way that the time at which the robot r_1 arrives at the sub-goal state serves as the end time. Planning is arranged so that, if the time is at the end time or a subsequent time, the robot r_1 remains at rest at the sub-goal state. The planning for robot r_1 may be carried out without taking the end time into consideration.

In other words, the sub-plan generation algorithm is as follows (Fig. 8):

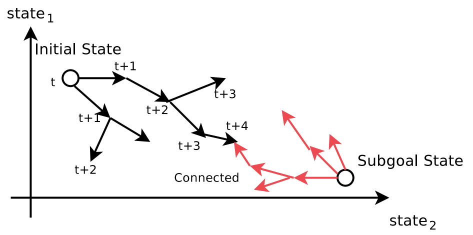
1) If the robot has the load (i.e., $r = r_1$) and its state is the sub-goal state and a non-collision state, sub-plan generation ends.
2) If the state is the sub-goal state and a non-collision state, and the end time has not been reached, that state is put into the sub-plan.
3) If the sub-goal state and non-collision state are in effect and the end time has been reached, sub-plan generation ends.
4) In any other state, planning toward the sub-goal is carried out (plan_toward_subgoal()).
5) Repeat from 1.

In the "plan_toward_subgoal()" routine (Fig. 10), planning towards a sub-goal state is carried out. This is realized with RRT using bidirectional search effectively.

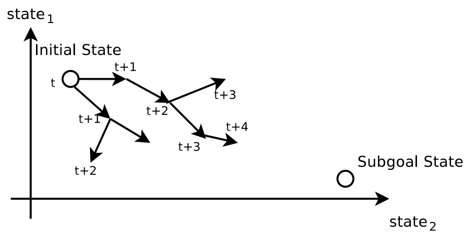
A search tree $T_{w/time}$ that includes a time parameter (in other words, a search for states in which the robot does not collide with other robots) grows from the state s_{init} at



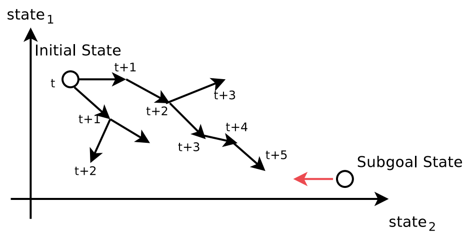
(a) The search trees extend from both sides.



(b) The search trees from both sides join together at a certain state.



(c) When the search trees are joined together, the search tree that does not include time is canceled.



(d) The search is continued.

Fig. 9. Bidirectional searching using a search tree that included time and using a search tree that did not include time (the example shows a projection of a two-dimensional space, with the time axis perpendicular to the page).

which the robot planning begins. At the same time, a search tree T_{w/o_time} that does not include a time (in other words, a search for the robot's movable areas in which no other robots are present) grows from the sub-goal state $s_{subgoal}$ of the robot (Fig.9(a)). Like the bidirectional search RRT, these grow toward a random-sampled state. If the two search trees are connected (Fig.9(b)), T_{w/o_time} is re-initialized (in other words, the results are destroyed before completion), and the search is continued (Fig.9(c)). $T_{w/time}$ is retained at that point. By doing this, the growth of the $T_{w/time}$ search tree that includes a time is guided by the T_{w/o_time} search tree

```
plan_toward_subgoal($r$, $t$, $s_r$)
  $T_{w/o\_time}$.init($s_{subgoal}$);
  for $k = 1$ to K do
    $s_{rand} \leftarrow$ RANDOM_STATE($r$);
    if not (EXTEND($T_{w/time}$, $s_{rand}$) = Trapped) then
      if is_reached($T_{w/time}$, $s_{subgoal}$) then
        return path($T_{w/time}$, $s_{subgoal}$);
      if (EXTEND($T_{w/o\_time}$, $s_{rand}$) = Reached) then
        $T_{w/o\_time}$.init($s_{subgoal}$);
    $s_{rand} \leftarrow$ RANDOM_STATE($r$);
    if not (EXTEND($T_{w/o\_time}$, $s_{rand}$) = Trapped) then
      if (EXTEND($T_{w/time}$, $s_{rand}$) = Reached) then
        $T_{w/o\_time}$.init($s_{subgoal}$);
      if is_reached($T_{w/time}$, $s_{subgoal}$) then
        return path($T_{w/time}$, $s_{subgoal}$);
    $s \leftarrow s_{subgoal}$;
    if (EXTEND($T_{w/time}$, $s$) = Reached) then
      return path($T_{w/time}$, $s_{subgoal}$);
  return Failure;
```

Fig. 10. Algorithm for planning toward a sub-goal state.

that does not include a time, which improves the searching efficiency. The algorithm is as shown in Fig.10.

In this study, the EXTEND operation was also used toward the sub-goal state in all cases except for bidirectional searching. This enables more efficient searching in cases where the robot can advance directly toward the sub-goal state.

C. The alignment of the end time of plans

After plans are generated for all sub-tasks, the end times of the plans are aligned to that of the longest one in robots. For this purpose, all plans are filled with non-collision state until reaching the end time of the longest one.

V. Experiments

In order to estimate the computational overhead for the proposed technique, we have compared the execution times using normal RRT with our technique. The environments shown in Figs. 2, 3 and 4 were used. The cooperation procedure employed was described in Table I. The computer used for the experiments was an Athlon XP 3200+ (2.2 GHz) with a Linux operating system. The number of retries for each sub-plan (the MAX_RETRY parameter in Fig.6) was set to be 5.

Table II shows the comparison result. We plotted one of the generated plans by the proposed technique in Fig. 11.

In order to apply the normal RRT (RRT-ConCon, Fig.1), goal states have to be explicitly provided for each of the sub-tasks. With that in mind, we randomly chose a pair of sub-goal positions for each of the sub-tasks from the "transfer locations". For the robots that are unrelated to a sub-task, we set sub-goal positions for them to stay at their current positions. The planning time was measured for each sub-task, and the total figures were listed in Table II. The maximum execution time of each of the sub-tasks is set to 10 minutes.

TABLE II

COMPARISON OF EXECUTION TIMES WITH THE PROPOSED TECHNIQUE AND WITH NORMAL RRT (RRT-CONCON). (THE PERCENTAGES IN PARENTHESES ARE RELATIVE TO THE RESULT USING NORMAL RRT AS THE STANDARD.)

	Execution time (sec)	
	Proposed method	normal RRT
Environment A	13.08 (254.2%)	5.15 (100.0%)
Environment B	41.52 (13.1%)	315.92 (100.0%)
Environment C	95.29 (17.9%)	533.24 (100.0%)

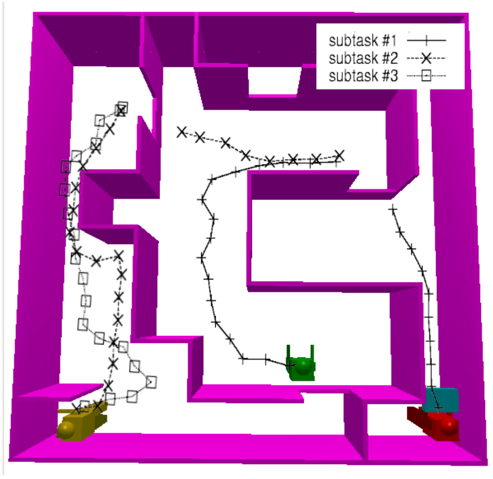


Fig. 11. One of generated plans by the proposed technique with environment B.

With Environment A, about twice of the execution time was required for the proposed technique, whereas with Environment B, the execution time of the proposed technique was shorter than that of the normal RRT. The execution time was about 87% less than that of normal RRT. With Environment C, the difference was about 82%.

VI. DISCUSSION

In section V, we compared the execution time with that of normal RRT (RRT-ConCon). With Environment A, it was about twice of normal RRT of the execution time for the proposed technique, but with Environment B and C, the execution time were very shorter than that of normal RRT. They were about 80% less than that of normal RRT.

There are two possible reasons about the overhead of the execution time of normal RRT. One reason may be that the search space was broader and more complicated in Environment B and C, so the problem in itself was more difficult. Also, with the normal RRT, sub-goals were provided for all of the robots, so the path searching included robots that were not involved in the task. Therefore, extra time was required in the execution of the normal RRT. Considering these results, the execution time for the proposed technique was longer than that with the normal RRT. However, we can confirm that the more difficult the problem, the more likely the overhead will be relatively small.

VII. CONCLUSION

In this paper, we proposed a planning algorithm using RRT to operate multiple robots cooperatively in a task. We tested the proposed technique in simulations, and have confirmed its effectiveness.

Further improvements in the search efficiency of the algorithm will be sought. According to Kuffner et al, probabilistic selection of the node for extension based on the area of its Voronoi region is proved to be effective [3]. By incorporating this result, the search performance can be improved in difficult problems.

Also, we intend to conduct experiments applying this algorithm to humanoid robots in a real environment.

REFERENCES

[1] J. Ota, N. Miyata, T. Arai, E. Yoshida, D. Kurabatashi, and J. Sasaki, "Transferring and regrasping a large object by cooperation of multiple mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems 95*, 1995.

[2] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions* (B. R. Donald, K. M. Lynch, and D. Rus, eds.), pp. 293–308, Wellesley, MA: A K Peters, 2001.

[3] J. Kuffner and S. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000), San Francisco, CA, April 2000.*, 2000.

[4] J. Kuffner, K. Nishikawa, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," in *Proc. 11th Int'l Symp. of Robotics Research (ISRR 2003)*, 2003.

[5] S. Kagami, J. Kuffner, K. Nishiwaki, K. Okada, M. Inaba, and H. Inoue, "Humanoid arm motion planning using stereo vision and rrt search," in *Proc. of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, pp. 2167–2172, 2003.

[6] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proc. of IROS-2002*, 2002.

[7] K. Okada, M. Inaba, and H. Inoue, "Walking navigation system of humanoid robot using stereo vision based floor recognition and path planning with multi-layered body image," in *Proc. of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2003.

[8] S. M. LaValle *et al.*, "Motion Strategy Library (MSL)." http://msl.cs.uiuc.edu/msl/.

[9] Y. Inoue, T. Tohge, and H. Iba, "Cooperative transportation by humanoid robots – learning to correct positioning –," in *Proc. of the Hybrid Intelligent Systems (HIS2003)*, pp. 1124–1133, 2003.