数理解析研究所講究録
1185 巻 2001 年 11-17

Distributed Motion Generation for Carrying a Ladder by Two Omni-Directional Robots

Yuichi Asahiro[1], Eric Chung-Hui Chang[2], Amol Mali[2], Syunsuke Nagafuji[1], Ichiro Suzuki[2], and Masafumi YAMASHITA[1]

[1]Dept. of Computer Science and Communication Engineering, Kyushu University
6–10–1 Hakozaki, Higashi-ku Fukuoka, 812-8581, Japan
{asahiro@, nagafuji@tcslab., mak@}csce.kyushu-u.ac.jp
[2] Dept. of Electrical Engineering and Computer Science
University of Wisconsin–Milwaukee, Milwaukee, WI 53201, U.S.A.
{ecchang, mali, suzuki}@cs.uwm.edu

Abstract: The problem of moving a pair of omni-directional robots carrying a ladder using distributed control is discussed. We first consider the case in which two robots that may differ only in their maximum speeds are situated in an obstacle-free workspace, and present two distributed algorithms. Next, a distributed algorithm is presented for the case in which the workspace is a narrow corridor with a 90 degree corner and two robots are chosen from a large pool of robots having different characteristics in terms of the maximum speed, path generation strategy, sensitivity to the motion of the other robot, etc. The effectiveness of the algorithms is evaluated using computer simulation.

Keywords: distributed control, mobile robots, transportation

1 Introduction

There are two general approaches for controlling multiple robots transporting an object. One is the centralized approach in which the motion of the robots is generated by an outside entity that can observe the global state of the system [7, 12]. The other is the distributed approach, where every individual robot has to decide its motion based on the local information available to it [8, 9]. This paper discusses the problem of transporting a ladder, or any long object such as a rocket and a bridge, using two omni-directional robots under distributed control.

Within the framework of the centralized approach, it is possible to compute a time-optimal motion of two such robots in an obstacle-free workspace using optimal control theory, under the assumption that the speed of the robots is either 0 or a given constant at any moment during a motion [5, 10]. Fig. 1 shows a instance of this problem in which robots located at A and B must move to A' and B', respectively, and a time-optimal motion for this instance obtained by this method is shown in Fig. 2. Unfortunately, the applicability of this method for actually moving physical robots in optimal time is somewhat limited, because (i) the method uses complex calculation involving elliptic integral, and (ii) physical robots cannot always execute a computed motion precisely — they can neither accelerate to the maximum speed instantaneously, nor move along a given trajectory precisely due to mechanical imprecision and the unpredictability of the environment (such as a slight incline of the floor).

In contrast, in the distributed approach the robots can cope with unexpected perturbation by continuously monitoring their progress and dynamically adjusting their trajectories. The overall motion resulting from such a distributed strategy can be nearly as efficient as an optimal motion [1, 2]. One should also keep in mind, however, that good distributed algorithms are usually much harder to design than centralized algorithms. For instance, the fact that the path of neither robot is straight in the optimal motion of Fig. 2 indicates that efficient motion may not be attainable distributively if either robot simply attempts to reach its destination as quickly as possible.

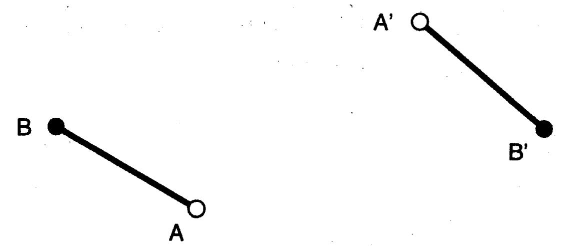

Figure 1: An instance of the ladder carrying problem.

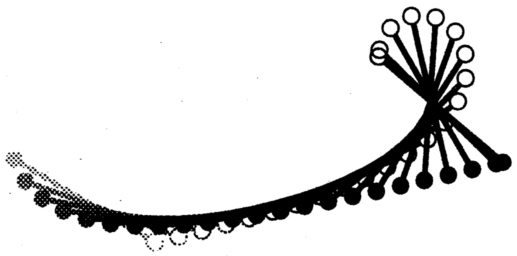

Figure 2: Time-optimal motion for the instance shown in Fig. 1.

A number of distributed algorithms for carrying a ladder using two identical omni-directional robots (hence having the same maximum speed) have been reported for the case in which there are no obstacles in the workspace [1, 2]. The goal of this paper is to consider the problem under the assumption that the robots are not necessarily identical and the workspace is not necessarily obstacle-free. Specifically:

1. We present two distributed algorithms for the case in which two robots possibly having different maximum speeds are situated in an obstacle-free workspace. The first algorithm, ALG1, assumes that each robot knows the maximum speeds of both robots, while the second algorithm, ALG2, is for the case this information is not available. The algorithms are evaluated using computer simulation.

2. We then present and evaluate an algorithm called ALG3 for the case in which two robots must transport a ladder through a corridor with a 90 degree corner. We assume that the two robots can have different characteristics in terms of the selection of a path through the corner, the maximum speed, "adherence" to stay on the intended path, and reaction to the other robot's motion.

Due to space limitation we are not able to present some of the details. The missing details may be found in the references, or will be reported in forthcoming papers.

2 The Model

The model of the robots we use is based on the omni-directional robots developed at RIKEN [3].

We represent each robot R as a disk. One end of the ladder is attached to a force sensor that we model as an ideal spring located at the center of R. At any time during a motion the vector from the center of R to the tip of the ladder attached to its force sensor is called the *offset vector*, and is denoted $\vec{o}$. The term *offset* refers to $|\vec{o}| = |(D-\ell)/2|$, where D is the distance between (the centers of) the two robots and ℓ is the length of the ladder.

An *algorithm* for robot R with maximum speed V is any procedure that computes a *velocity vector* $\vec{v}$ with $|\vec{v}| \leq V$, using available information such as the robots' current and final positions, the offset vector $\vec{o}$, and the geometry of the workspace. We assume that R repeatedly computes $\vec{v}$ and moves to a new position with velocity $\vec{v}$ for unit time.

We evaluate our algorithms by computer simulation. For simplicity we use discrete time and assume that both robots compute their respective velocity vectors and move to their new positions at time instants $0, 1, \ldots$.

3 Obstacle-Free Workspace

Consider two robots A and B in an obstacle-free workspace with respective goal positions A' and B'. A and B may have different maximum speeds. For convenience of discussion we set up a Cartesian coordinate system as shown in Fig. 3, where A and A' are at the origin $(0,0)$ and $(L,0)$, respectively, and α and β are the angles that AB and $A'B'$ make with the x-axis, respectively. In this section, we assume $L' \leq L$ and $0 \leq \alpha, \beta \leq 180°$, where $L = |AA'|$ and $L' = |BB'|$. Minor changes are needed in the following discussion for other cases.

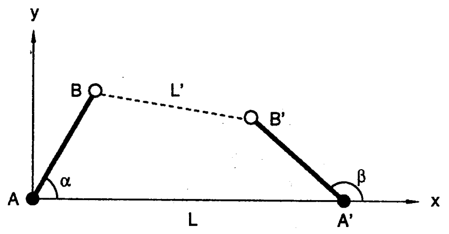

Figure 3: The setup for ALG1 and ALG2.

3.1 Distributed algorithms ALG1 and ALG2

In this section we introduce two distributed algorithms ALG1 and ALG2. ALG1 is for the case where the two robots' maximum speeds are known to both robots, while ALG2 is for the case where neither robot knows the other robot's maximum speed. In either algorithm we assume that the robots' current and goal positions are known to both robots. (This assumption can be expensive to realize in practice.)

Both algorithms are memoryless in the sense that their output is a function of the current state (and is independent of the motions in the past). It is therefore sufficient to view A and B of Fig. 3 as the robots' current positions and specify how the velocity vector is computed from A, B, A' and B'.

Algorithm ALG1
Both robots know the maximum speeds v_A and v_B of both, as well as the positions A, B, A', B', and hence angles α, β and their offset vectors $\vec{o}_A$ and $\vec{o}_B$. c_1, c_2, and $s \geq 0$ are some constants. Since both robots have the same information we explicitly describe the procedure for both.

Step 1: Let $e_A = L/v_A$ and $e_B = L'/v_B$. Let $\vec{t}_A$ and $\vec{t}_B$ be vectors directed from A to A' and from B to B', respectively, such that

- $|\vec{t}_A| = 1$ and $|\vec{t}_B| = (e_B/e_A)^{c_1}$ if $e_A \geq e_B$, and
- $|\vec{t}_A| = (e_A/e_B)^{c_1}$ and $|\vec{t}_B| = 1$ if $e_A < e_B$.

Step 2: Let $\vec{r}_A$ and $\vec{r}_B$ be "rotational vectors" of length $c_2(\beta - \alpha)$ that rotate the ladder counterclockwise, whose directions are $\alpha + \pi/2$ and $\alpha - \pi/2$, respectively.

Step 3: Scale the offset vectors as $\vec{h}_A = s\vec{o}_A$ and $\vec{h}_B = s\vec{o}_B$.

Step 4: $\vec{T}_A = \vec{t}_A + \vec{r}_A + \vec{h}_A$ and $\vec{T}_B = \vec{t}_B + \vec{r}_B + \vec{h}_B$.

Step 5: Output velocity vector

- $\vec{v}_A = v_A\vec{T}_A/\max\{|\vec{T}_A|, |\vec{T}_B|\}$ for robot A, and
- $\vec{v}_B = v_B\vec{T}_B/\max\{|\vec{T}_A|, |\vec{T}_B|\}$ for robot B.

In **Step 1** ALG1 tries to slow the robot that would otherwise reach its goal sooner than the other according to the estimates e_A and e_B. The constant s used in **Step 3** is a parameter indicating how a robot reacts to the offset.

Algorithm ALG2
The robots have all the information available in ALG1, except robot A does not know v_B and robot B does not know v_A. We use additional constants v'_A and v'_B.

Step 1: Robot A runs ALG1 using v_A and v'_B in place of v_A and v_B, respectively. Likewise robot B runs ALG1 using v'_A and v_B. Let $\vec{v}'_A$ and $\vec{v}'_B$ the velocity vectors obtained.

Step 2: Output velocity vector

- $\vec{v}_A = \vec{v}'_A$ if $|\vec{v}'_A| \leq v_A$, and $\vec{v}_A = v_A\vec{v}'_A/|\vec{v}'_A|$ otherwise, for robot A.
- $\vec{v}_B = \vec{v}'_B$ if $|\vec{v}'_B| \leq v_B$, and $\vec{v}_B = v_B\vec{v}'_B/|\vec{v}'_B|$ otherwise, for robot B.

In ALG2 we use v'_A and v'_B as an estimate of unknown v_A and v_B, and the output of ALG2 coincides with that of ALG1 if $v'_A \leq v_A$ and $v'_B \leq v_B$. We assume that v'_A and v'_B are constants supplied to ALG2, since it is one of our basic goals to keep the algorithms memoryless (memoryless algorithms can tolerate a finite number of transient errors). It would be interesting, however, to modify ALG2 so that the robots will choose suitable values for v'_A and v'_B based on their recent history.

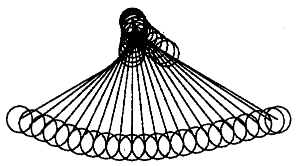

Figure 4: ALG1 for $k = 1$, $\alpha = 30°$, $L = 200$.

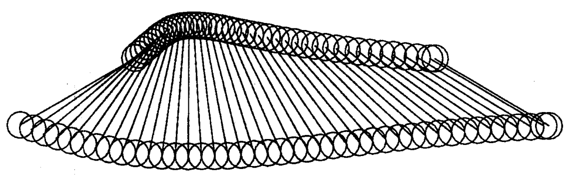

Figure 5: ALG1 for $k = 1$, $\alpha = 30°$, $L = 400$.

3.2 Experimental results by computer simulation

Using the setup of Fig. 3 we evaluate the performance of the algorithms in terms of the time necessary for the robots to reach their goals. The length ℓ of the ladder is 100, and the radius of the disks representing the robots is 10. We use $v_A = 1$ and $v_B = k$ (so k is the ratio of v_B to v_A), and in the following discuss mainly the results for the case $k \geq 1$. The parameters c_1, c_2 and s of ALG1 are set to 3, 0.5, and 0.1, respectively, that have been found to work well when $v_A = v_B$ [1]. To reduce the number of instances to examine, we experiment with only two values of L, $L = 200$ $(= 2\ell)$ and 400 $(= 4\ell)$, while changing α in the range from 0° to 90° and setting $\beta = 180° - \alpha$.

Fig. 4 and Fig. 5 show the motions generated by ALG1 for $k = 1$ and $\alpha = 30°$, for $L = 200$ and $L = 400$, respectively. Fig. 6 and Fig. 7show the same, for $k = 3$ instead of 1. The finish times for $L = 200$ are 205 ($k = 1$) and 201 ($k = 3$), and for $L = 400$ they are 404 ($k = 1$) and 401 ($k = 3$). Note that in Fig. 6 and Fig. 7 robot B initially slows down considerably, allowing robot A to rotate more quickly than in Fig. 4 and Fig. 5.

Fig. 8 and Fig. 9 show the finish times for $k =$ 1, 2, 3, 4 and 5, for $L = 200$ and 400, respectively.

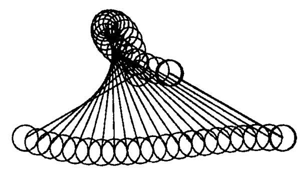

Figure 6: ALG1 for $k = 3$, $\alpha = 30°$, $L = 200$.

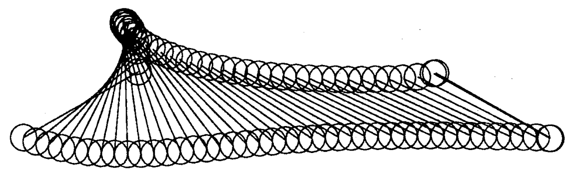

Figure 7: ALG1 for $k = 3$, $\alpha = 30°$, $L = 400$.

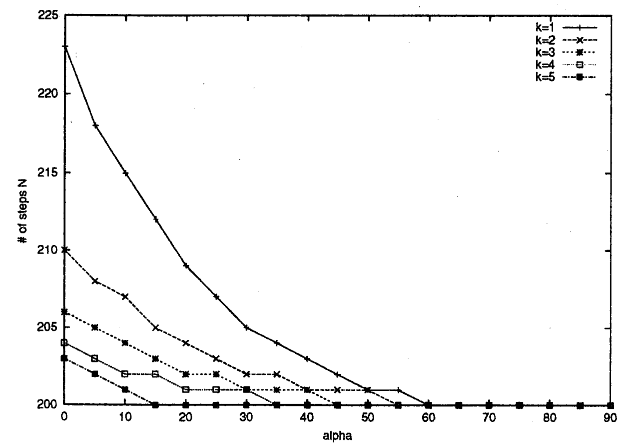

Figure 8: Finish times of ALG1 for $k = 1, 2, 3, 4$ and 5, for $L = 200$.

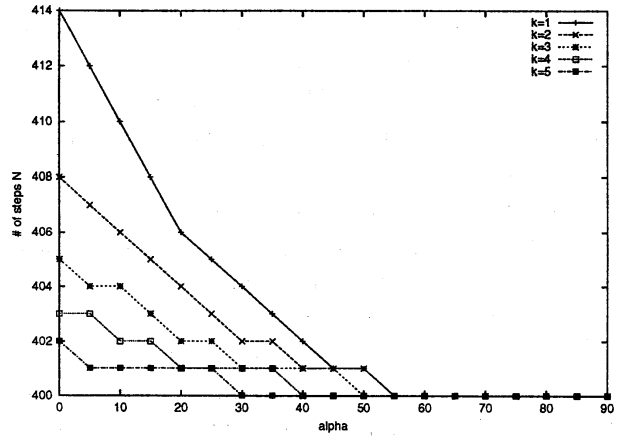

Figure 9: Finish times of ALG1 for $k = 1, 2, 3, 4$ and 5, for $L = 400$.

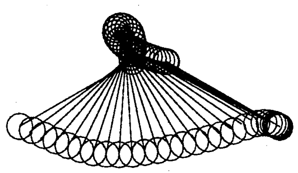

Figure 10: ALG2 using an estimate $k^* = 3$ in place of $k = 1/3$.

We observe that the time decreases as k increases when $\alpha \leq 60°$. This phenomenon is quite natural, since smaller α implies more necessary rotation, and larger k causes B to slow down, thus allowing A to rotate more quickly.

Fig. 10 shows a motion generated by ALG2 for $k = 1/3$, $\alpha = 30°$ and $L = 200$, where (for simplicity) both robots use an estimate of $k^* = 3$ in place of the unknown k. (The estimates of k by A and B may differ in real situations.) Note that the robots successfully complete the task even though their estimate k^* is not at all close to actual k. However, the large discrepancy between k^* and k has resulted in a noticeable decline in the performance in terms of the finish time — the finish time of ALG2 in Fig. 10 is almost 28% larger than that of ALG1 in Fig. 11 for the same instance.

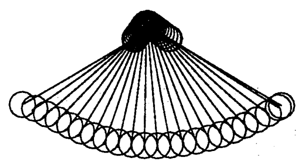

Figure 11: ALG1 with $k = 1/3, \alpha = 30°, L = 200$.

4 Corridor with a Corner

The distributed approach works well also for the case in which the robots must go through a 90 degree corner in a corridor avoiding both robot-to-wall and ladder-to-wall collision. The motion shown in Fig. 12 has been obtained by an algorithm that is similar in spirit to the ones presented in the preceding section, with an additional step in which each robot computes a target path through the corner before starting the motion. During the motion each robot attempts to move along the path while adjusting its positions based on both the motion of the other robot observed through the force sensor and the need to prevent collision. We assume that the robots can detect how close they and the ladder are to the walls, from their positions and the geometry of the workspace.

While the robots in the preceding section are assumed to be identical in all aspects other than their maximum speed, in this section we assume that they may differ in a number of other characteristics as well — (i) selection of a path through the corner (**in**, **middle** or **out**), (ii) the maximum speed (**high** or **low**), (iii) adherence to the intended path (**high** or **low**), and (iv) reaction to the motion of the other robot (**high**, **low**, or **nonlinear**). These variations result in a total of 36 different types of robots from which two are chosen to carry out the task. We assume that the robots do *not* know the characteristics of the other robot.

First, robot R generates a path P through the corner as a Kochanek-Bartels spline [6] that travels either close to the inner walls of the corner (**in**), close to the outer walls of the corner (**out**), or somewhere in between the two (**middle**).

The remaining characteristics of R are deter-

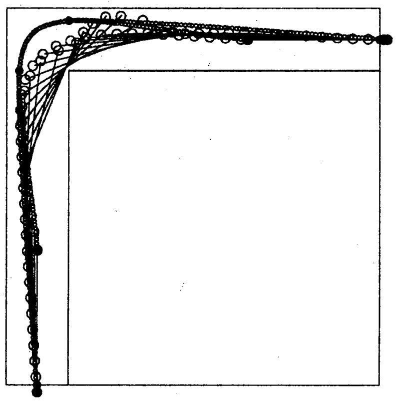

Figure 12: Motion through a corner generated by ALG3. The robots start at the bottom of the figure and makes a right turn. The intended paths of the robots are shown in small circles.

mined by the following algorithm ALG3 that R uses to compute its velocity vector. It is assumed that the offset $|\vec{o}|$ can be as large as the the radius 10 of the disk representing R.

Algorithm ALG3

Step 1: $\vec{u} = (1/10)(\vec{g} + f(\vec{o}) + \vec{c})$, where

- $\vec{g}$ is a vector of size 10 directed from the center of R toward the "current target position",
- $\vec{o}$ is the offset vector and f is a function that converts $\vec{o}$ into another vector such that $|f(\vec{o})| \leq 10$ (see details below), and
- $\vec{c}$ is a suitable "correction vector" needed to prevent collision. (We omit the details of $\vec{c}$.)

The factor $1/10$ effectively reduces the sizes of $\vec{g}$ and $f(\vec{o})$ to within 1.

Step 2: $\vec{w} = \vec{u}$ if $|\vec{u}| \leq 1$, and $\vec{w} = \vec{u}/|\vec{u}|$ if $|\vec{u}| > 1$. That is, $\vec{w}$ is the result of "clamping" vector $\vec{u}$ at length 1 so that $|\vec{w}| \leq 1$.

Step 3: Output the velocity vector $\vec{v} = V\vec{w}$, where V is the maximum speed of R.

In our experiments the maximum speed V of R can be either 2 (high) or 1 (low).

A robot with high adherence attempts to return to P more quickly than one with low adherence when it deviates from P. We control the adherence of R by choosing vector $\vec{g}$ in **Step 1** appropriately: If adherence is high then the "current target position" (at which $\vec{g}$ is aimed) is chosen to be a point on P relatively close to the robot's current location. If adherence is low then $\vec{g}$ is directed toward a point on P that is farther away. (We omit the details of how such points are actually chosen in our experiments.)

Function f of **Step 1** determines how R reacts to the motion of the other robot observed through $\vec{o}$. We use the following three variations.

1. high: $f(\vec{o}) = \vec{o}$. The robot is highly sensitive to $\vec{o}$.
2. low: $f(\vec{o}) = 0.5\vec{o}$. The robot's sensitivity is low.
3. nonlinear: $f(\vec{o})$ equals the zero vector $\vec{0}$ if $|\vec{o}| \leq 5$, and $2\vec{o} - \vec{o}/|\vec{o}|$ if $|\vec{o}| > 5$. The robot reacts to $\vec{o}$ only after its magnitude exceeds 5.

Note that $|f(\vec{o})| \leq |\vec{o}| \leq 10$ holds in all three cases.

The motion shown in Fig. 12 is obtained by ALG3 where both robots use the same out path and has the same low adherence. The robot in front has high maximum speed and high reaction (f), while the other robot has low maximum speed and nonlinear reaction.

To evaluate ALG3 we randomly generate 100 pairs of robots from the pool of 36 robots and examine the probability that the task is completed successfully (a motion is considered unsuccessful if the offset exceeds 10). The width of the corridor is 200 and the ladder length ℓ varies between 400 and 480. (The robots always fail when $\ell = 490$.) The success rate decreases from 100% for $\ell = 400$ to 75% for $\ell = 480$ if both robots are allowed to choose a path that is either in, middle or out, while the rate increases to 100% for $\ell = 480$ if neither is allowed to choose an in path. However, if we allow the robots' adherence to be even higher (than high), then the rate drops again to 57% for $\ell = 480$ even without in paths. The reader is referred to [4] for additional results and a detailed analysis of the effect of these parameters to the overall performance.

5 Concluding Remarks

We have presented three distributed algorithms for two robots carrying a ladder under various conditions, and evaluated their performance using computer simulation. We are currently working on a detailed analysis of ALG3.

References

[1] Y. Asahiro, H. Asama, S. Fujita, I. Suzuki and M. Yamashita, "Distributed algorithms for carrying a ladder by omnidirectional robots in near optimal time," in *Sensor Based Intelligent Robots*, H.I. Christensen, H. Bunke and H. Noltemeier, Eds., Lecture Notes in Artificial Intelligence, Vol. 1724, Springer Verlag, Heidelberg, Germany, pp. 240–254, December 1999.

[2] Y. Asahiro, H. Asama, I. Suzuki and M. Yamashita, "Improvement of distributed control algorithms for robots carrying an object," in *Proceedings of 1999 IEEE International Conference on Systems, Man and Cybernetics*, pp. VI 608–613, October 1999.

[3] H. Asama, M. Sato, L. Bogoni, H. Kaetsu, A. Matsumoto, and I. Endo, "Development of an omni-directional mobile robot with 3 DOF decoupling drive mechanism," *Proc. IEEE Int. Conf. on Robotics and Automation,* pp. 1925–1930, 1995.

[4] E.C.-H. Chang, "Distributed motion coordination of two robots carrying a ladder in a corridor," MS Thesis, EECS Department, University of Wisconsin–Milwaukee, in preparation.

[5] Z. Chen, I. Suzuki, and M. Yamashita, "Time optimal motion of two robots carrying a ladder under a velocity constraint," *IEEE Trans. Robotics and Automation 13*, 5, pp. 721–729, 1997.

[6] D. Hearn and M.P. Baker, *Computer Graphics C Version*, Prentice Hall, 1996.

[7] Y. Kawauchi, M. Inaba and T. Fukuda, "A principle of distiributed decision making of cellular robotic system (CEBOT)," *Proc. IEEE Int. Conf. on Robotics and Automation,* pp. 833-838, 1993.

[8] K. Kosuge and T. Oosumi, "Decentralized control of multiple robots handling and objects," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems,* 556-561 (1996).

[9] N. Miyata, J. Ota, Y. Aiyama, J. Sasaki, and T. Arai, "Cooperative transport system with regrasping car-like mobile robots," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems,* pp. 1754–1761, 1997.

[10] S. Nagafuji, Y. Asahiro, M. Yamashita and I. Suzuki, "Time-optimal motion of two heterogeneous robots carrying a large object," in *Proceedings of 1999 Joint Conference of Electrical and Electronics Engineers of Kyushu* (in Japanese), page 742, October 1999.

[11] J. Pearl, *Heuristics – Intelligent Search Strategies for Computer Problem Solving,* Addison-Wesley, 1984.

[12] Z. Wang, E. Nakano and T. Matsukawa, "Realizing cooperative object manipulation using multiple behavior-based robots," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems,* pp. 310–317, 1996.