

A Distributed Ladder Transportation Algorithm for Two Robots in a Corridor*

Yuichi Asahiro[†] Eric Chung-Hui Chang[‡] Amol Mali[‡] Ichiro Suzuki[‡] Masafumi Yamashita[†]

[†] Department of Computer Science and Communication Engineering, Kyushu University, Fukuoka, Japan

[‡] Department of Electrical Engineering and Computer Science, University of Wisconsin–Milwaukee, Milwaukee, U.S.A.

Abstract

We consider the problem of transporting a long object, such as a ladder, through a 90 degree corner in a corridor using two omni-directional robots that do not necessarily have identical characteristics. A distributed algorithm is presented in which each robot computes its own motion based on the current and goal positions of the ladder, the locations of the walls, and the motion of the other robot observed indirectly through the link between the robot and the ladder. We evaluate the performance and robustness of the algorithm using extensive computer simulation by changing several parameter values that affect the key characteristics of the robots, including the maximum speed, the guide path through the corner, and sensitivity and reaction to the motion of the other robot. The simulation results indicate that if the parameter values are chosen within certain reasonable ranges, then overall the algorithm works quite well even for robots having different characteristics. It is also shown that the robustness of the algorithm critically depends on the differences between the robots in the values of two parameters.

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 (e.g., [10]). 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].

In this paper we investigate the problem of transporting a ladder, or any other long object such as a rocket or a bridge, using two omni-directional robots under distributed control. There can be many variations of this problem – the robots may be identical or different, the workspace may be obstacle-free or it may contain obstacles, and so on. For many of these variations, the distributed approach may be advantageous or even necessary.

For instance, for the case of an obstacle-free workspace, while it is possible to use a centralized approach to compute a time-optimal motion for two robots [6], a distributed approach can better cope with unexpected perturbations since the robots continuously monitor their progress and dynamically adjust their trajectories. It has been reported that the overall motion resulting from such a distributed



Figure 1: RIKEN’s omni-directional robots carrying a bar (ladder).

strategy can be nearly as efficient as an optimal solution [1, 2]. If the workspace includes (possibly moving) obstacles and the robots can detect only the obstacles immediately ahead, a distributed approach may again be better since robots under centralized control may not be able to react to the changing environment quickly enough.

Another case is where there is a group of robots with different characteristics — the robots may have different maximum speeds, react differently to the motion of the other robot, or use slightly different planning strategies [3] — and we wish to be able to select any two robots from the group to successfully carry a ladder to its destination. Indeed, since no two physical robots are truly identical, practical robot algorithms should be tolerant against variations in the robots’ actions. From this viewpoint, a distributed approach may be far more advantageous, because we do not know which two robots may be selected, and it would be difficult to plan in advance for all possibilities using a centralized approach.

Given these motivations, this paper considers the ladder transportation problem for two robots, for the case where the workspace is a corridor with a 90 degree turn and the robots can differ in several characteristics. We examine the feasibility of the distributed approach by designing a distributed algorithm and evaluating its performance through extensive computer simulation. Specifically: (1) We create a fairly accurate model of a robot-ladder system based on actual omni-directional robots developed at RIKEN (The Institute of Physical and Chemical Research, Japan) [4] (Fig. 1). The model has several parameters for changing certain key characteristics of the robots. (2) We examine the factors affecting the performance of the al-

*Supported in part by a Scientific Research Grant-in-Aid from the Ministry of Education, Science and Culture in Japan. Communicating author: I. Suzuki, suzuki@cs.uwm.edu

gorithm using two criterion, namely the distance travelled by the robots and the number of steps taken to go through the corner. (3) We examine the robustness of the algorithm in terms of the success rate of a large number of randomly generated pairs of robots having various characteristics. For the pairs that fail, we examine the cause of the failure (whether the robots get stuck in the corner, or the linkage between the robots and the ladder breaks). (4) We examine how tolerant the algorithm is to differences in the characteristics of the two robots.

The simulation results indicate that if the parameter values are chosen within certain reasonable ranges, then overall the algorithm works quite well even for robots having different characteristics. Two parameters, however, if they differ widely between the robots, can cause the algorithm to fail frequently due to linkage breaks.

The rest of the paper is organized as follows. In Section 2 we present the model of the robot-ladder system. In Sections 3 and 4 we present the algorithm, describe its run-time parameters for modeling robots with various characteristics, and explain how we conduct computer simulation and evaluate the results. In Section 5 we analyze the simulation results. Conclusions are found in Section 6.

2 The Model

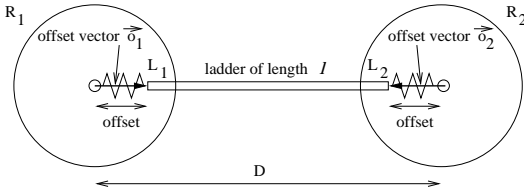


Figure 2: The robot-ladder system.

Fig. 2 shows the model of the robot-ladder system consisting of a ladder, represented as a straight line segment L_1L_2 , and two robots R_1 and R_2 , each represented as a disk. At RIKEN the robots are attached to the ends of the ladder through a flexible force sensor located at their centers. We model the sensor as an ideal spring, and let sc_i be the spring constant of R_i , $i = 1, 2$. Let ℓ be the length of the ladder, and D the distance between the centers of the robots. Whenever $D \neq \ell$ during motion there is a gap, or *offset*, between the center of R_i and L_i , and we call the vector from R_i 's center to L_i the *offset vector* \vec{o}_i . In order to reach equilibrium, the force exerted by R_1 's spring must be equal to the force exerted by R_2 's spring. So the magnitudes of the offset vectors are:

$$|\vec{o}_1| = \frac{sc_2}{sc_1 + sc_2} \times (D - \ell)$$

$$|\vec{o}_2| = \frac{sc_1}{sc_1 + sc_2} \times (D - \ell)$$

We assume that the link between R_i and the ladder breaks if the offset exceeds a given constant $olim_i$, that we call the *offset limit*.

Fig. 3 shows the workspace. It is a corridor with a 90 degree right turn that is 200 units wide, 2400 units long along the outer wall and 2000 units long along the inner

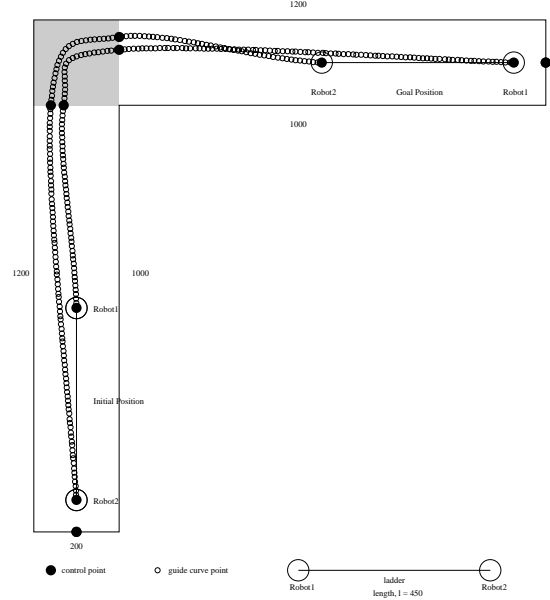


Figure 3: Corridor with a 90 degree right turn, initial and goal positions, guide paths, and control points.

wall. In our experiments the initial and goal positions of the ladder are parallel to and right down the middle of the corridor, with R_1 in front of R_2 .

3 The Algorithm

3.1 Computing a Guide Path

Each robot first computes, individually and without taking motion coordination into account, a *guide path* between its initial and goal positions through the corner that it later tries to follow during motion. We represent guide paths as a sequence of equally spaced points, as shown in Fig. 3. (In our experiments there are 160 points on each path.)

We chose to generate such paths as a cardinal spline [7] using several control points, though other splines can also be used. The control points are shown as black dots in Fig. 3, of which two near the corner (one at the “entrance” and another at the “exit” of the corner) determine where the path goes through in the corner. We use a parameter α_i , $0 \leq \alpha_i \leq 1$, to control the positions of these two control points for robot R_i . When α_i is closer to 0 the two control points are closer to the inner walls of the corner, and consequently the path takes a more inner route through the corner. On the other hand, when α_i is closer to 1 the path takes a more outer route. In Fig. 3 R_1 's path ($\alpha_1 = 0.65$) takes a slightly more inner route than that of R_2 ($\alpha_2 = 0.8$).

3.2 Motion through the Corridor

Each robot then begins its motion by repeatedly executing a *step* until both robots reach their goal positions, where in every step robot R_i computes a *motion vector* \vec{m}_i , and moves from its current position, say p_i , to a new location

$p_i + \vec{m}_i$, $i = 1, 2$. The direction and magnitude of \vec{m}_i depend on four often conflicting factors — the robot's attempt to (1) follow its guide path, (2) cooperate with the other robot to rotate the ladder towards the final orientation, (3) reduce the offset to prevent the linkage from breaking, and (4) prevent robot-wall and ladder-wall collision. Consequently, usually the robots cannot always stay on their guide paths.

To compute \vec{m}_i , we combine three vectors — the *target vector*, the *rotation vector*, and the *correction vector* — and then modify the result taking into account collision avoidance.

3.2.1 Target, Rotation, and Correction Vectors

The target vector \vec{t}_i , is a unit vector (i.e., of length 1) representing R_i 's attempt to follow its guide path. See Fig. 4. The vector starts at the center of R_i and points towards a point on the guide path towards which R_i wishes to move. We use a run-time parameter $lookahead_i$ to specify this point. Specifically, R_i first finds the point on the guide path closest to its current position, and then looks forward along the path the specified number of points to locate that point.

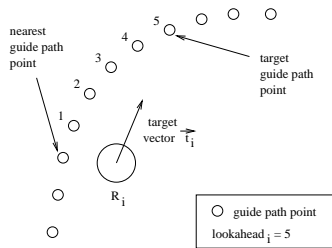


Figure 4: Goal vector pointing towards lookahead guide path point.

The rotation vector \vec{r}_i represents R_i attempt to slowly rotate the ladder from its initial orientation to its final orientation at the goal position. See Fig. 5. The rotation vectors of robots R_1 and R_2 are both perpendicular to the ladder and opposite in directions. When determining the magnitude of the rotation vector we attempt to spread the rotation evenly throughout the remaining motion of the ladder. Specifically, we approximate the distance that R_i has to travel to complete the rotation by the length of the arc, denoted *arclength*, that each tip of the ladder has to traverse before the goal orientation is reached. Using *arclength* and $distgoal_i$, which is the distance between the current and goal positions of R_i , we set

$$|\vec{r}_i| = k_{r,i} \times \frac{arclength}{distgoal_i},$$

where $k_{r,i}$, called the *rotation vector factor* of R_i , is a constant that enables us to control and experiment with the magnitude of the rotation vector. For instance, we can turn R_i 's rotation vector off by setting $k_{r,i}$ to 0. Note that in the above formula, we are using the magnitude of the target vector $|\vec{t}_i| = 1$ as a rough estimation of how far R_i would travel towards its goal in a single step.

The correction vector \vec{c}_i represents R_i 's attempt to reduce the offset. See Fig. 6. The vector starts at the center

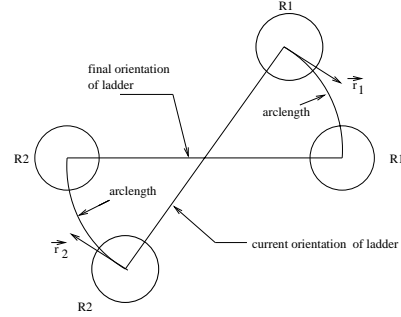


Figure 5: Rotation vectors \vec{r}_1 and \vec{r}_2 for turning the ladder toward the goal orientation.

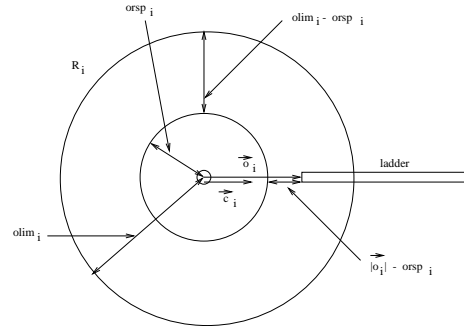


Figure 6: Correction vector \vec{c}_i for reducing the offset.

of R_i points towards the tip L_i of the ladder. The magnitude of \vec{c}_i is determined as follows.

First, observe that we do not necessarily want to have a nonzero correction vector whenever there is an offset. We may wish to allow R_i to move freely disregarding the offset until its magnitude $|\vec{o}_i|$ exceeds a user-definable scalar threshold $orsp_i$, called the *offset resistance starting point*, where $0 \leq orsp_i \leq olim_i$. For instance, if $orsp_i = 0.5 \times olim_i$, then there is no correction vector until $|\vec{o}_i|$ exceeds 0.5 of $olim_i$. Once $|\vec{o}_i|$ exceeds $orsp_i$, we increase the magnitude of the correction vector $|\vec{c}_i|$ linearly as $|\vec{o}_i| - orsp_i$ increases. To reduce the chances of link breakage, $orsp_i$ should not be too close to $olim_i$ — in our experiments $orsp_i$ is usually no larger than 0.5 of $olim_i$.

To summarize, if $|\vec{o}_i| \leq orsp_i$ then $\vec{c}_i = \vec{0}$; otherwise

$$\vec{c}_i = k_{c,i} \times \frac{|\vec{o}_i| - orsp_i}{olim_i - orsp_i} \times \frac{\vec{o}_i}{|\vec{o}_i|},$$

where $0 < k_{c,i} \leq 1$ is a constant, called the *correction vector factor* of R_i , that controls the magnitude of \vec{c}_i in relation to $|\vec{o}_i| - orsp_i$. Note that the maximum possible magnitude of \vec{c}_i is 1, achieved when $k_{c,i} = 1$ and the offset has reached the offset limit, i.e., $|\vec{o}_i| = olim_i$.

3.2.2 Combining the Three Vectors

We combine \vec{t}_i , \vec{r}_i , and \vec{c}_i into motion vector \vec{m}_i by

$$\vec{m}_i = v_{max,i} \times \text{CLAMP} \left(\frac{\vec{t}_i + \vec{r}_i}{|\vec{t}_i + \vec{r}_i|} + \vec{c}_i \right),$$

where $\text{CLAMP}(\vec{v})$ clamps vector \vec{v} at magnitude 1, i.e., $\text{CLAMP}(\vec{v}) = \vec{v}$ if $|\vec{v}| \leq 1$; otherwise $\text{CLAMP}(\vec{v}) = \vec{v}/|\vec{v}|$. $v_{max,i}$ is R_i 's maximum speed, or the largest distance it can cover in one step.

We normalize $\vec{t}_i + \vec{r}_i$ before adding \vec{c}_i for the following reason. The magnitude of \vec{c}_i reaches its maximum of 1 (assuming $k_{c,i} = 1$) when the offset of R_i is at the offset limit $olim_i$. At that moment the linkage between R_i and the ladder is about to break, and in such critical moments we want the magnitude of \vec{c}_i to be no smaller than that of the sum of the other two vectors so that the offset can be reduced. Normalizing $\vec{t}_i + \vec{r}_i$ ensures this.

Once the three vectors are combined we clamp the result at magnitude 1 to ensure that R_i does not move more than distance $v_{max,i}$ in a single step.

3.2.3 Modifying \vec{m}_i to Avoid Collision

We briefly explain how vector \vec{m}_i is modified when collision is imminent.

If R_i finds that it is about to collide with a wall, then it resolves \vec{m}_i into two orthogonal components and nullifies the one aimed towards the wall, as shown in Fig. 7(a).

Ladder-wall collision avoidance is attempted in much the same way. As shown in Fig. 7(b), if R_i finds that the ladder is about to collide with the inner walls of the corridor, then it resolves \vec{m}_i into two orthogonal components and nullifies those aimed either toward the inner walls or away from the corner.

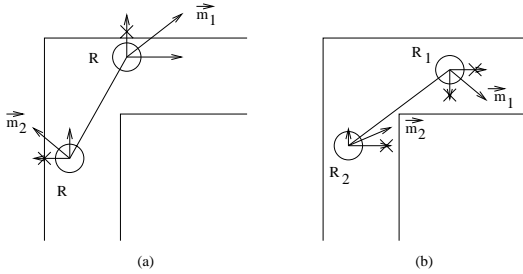


Figure 7: Avoiding (a) robot-wall collision and (b) ladder-wall collision. Crossed out components are nullified.

4 Parameters, Evaluation Criterion, Simulation Setup

Almost all parameters are user-definable at runtime so that we can experiment with robots having various characteristics. Those that are most relevant to our study, together with their sample values, are the following.

ladder length ℓ	450
R_1 offset limit $olim_1$	10
R_2 offset limit $olim_2$	10
R_1 offset resistance starting point $orsp_1$	0
R_2 offset resistance starting point $orsp_2$	0
R_1 correction vector factor $k_{c,1}$	1.0
R_2 correction vector factor $k_{c,2}$	1.0
R_1 maximum speed $v_{max,1}$	2
R_2 maximum speed $v_{max,2}$	2
R_1 spring constant sc_1	0.8
R_2 spring constant sc_2	1.0

R_1 rotation vector factor $k_{r,1}$	0.0
R_2 rotation vector factor $k_{r,2}$	0.0
R_1 lookahead $lookahead_1$	16
R_2 lookahead $lookahead_2$	16
R_1 α_1	0.65
R_2 α_2	0.8

The result of simulation using these values is shown in Fig. 8, where large circles represent snapshots of the robots during the motion.

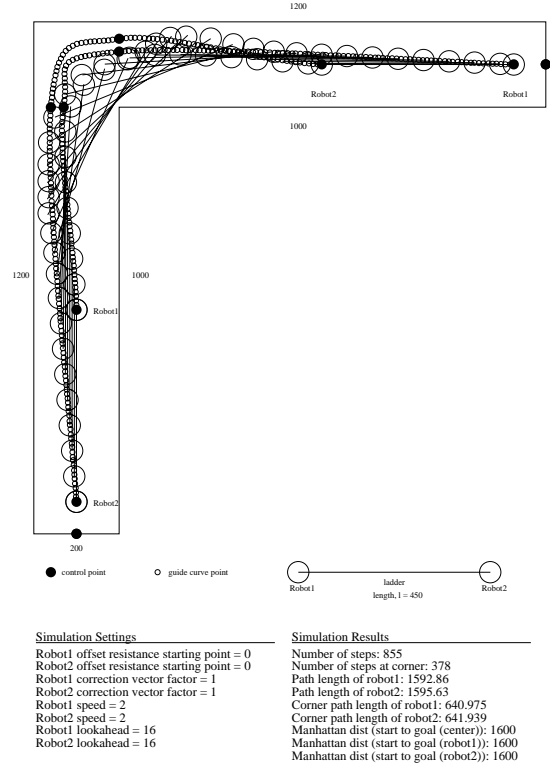


Figure 8: Simulation result showing robots successfully moving a ladder to the goal.

The statistics shown in the figure indicate that the robots successfully carry a ladder of length 450 to the goal in 855 steps, including 378 steps in the corner. (The ladder is considered to be *in* the corner when it intersects the 200×200 square region of the corner shown shaded in Fig. 3.) The lengths of the paths traversed by R_1 and R_2 are 1592.86 and 1595.63 units, respectively. The lengths of the paths when the ladder is in the corner are 640.975 and 641.939 units, respectively, for the two robots.

In another run not shown here due to space limitation, the robots successfully carry a ladder of length 485, which is close to the physical limit of what can be carried through the corner. The success for this difficult case is an indication of the basic soundness of the algorithm.

The method we adopted for simultaneously (1) evaluating the algorithm's performance and robustness when robots can have different characteristics, and (2) analyzing the effects of individual parameters or a set of parameters, was to randomly generate a large number of pairs of robots having various characteristics while setting parameters of

interest to specific values, and run these robots through the corner and collect statistics. We evaluate the algorithm's performance based on the number of steps and the robots' path lengths while the ladder is in the corner. For robustness, the criteria is the number of pairs of robots (from the large sample) that successfully carry the ladder to the goal. If the robots fail to reach the goal by either getting stuck in the corner or breaking the linkage, then we shall analyze possible causes of the failure. Specifically:

1. We generate a set S of 100 pairs of robots having various characteristics, randomizing the following parameters in reasonable ranges (shown in parenthesis): offset resistance starting point $orsp_1, orsp_2$ (0 to 0.2 of $olim_i$); maximum speed $v_{max,1}, v_{max,2}$ (0.5 to 1.5); lookahead $lookahead_1, lookahead_2$ (0 to 20); α_1, α_2 (0.35 to 0.9). Note that because of randomization each pair is likely to consist of two non-identical robots.

When generating S we set correction vector factors $k_{c,1}$ and $k_{c,2}$ to 1.0, spring constants sc_1 and sc_2 to 1.0, and rotation vector factors $k_{r,1}$ and $k_{r,2}$ to 0, instead of randomizing them. This is because we had already found the following by preliminary experiments: (i) Differences in spring constants together with differences in maximum speeds between the two robots can result in too many failures (the robot with the higher spring constant and higher maximum speed will be insensitive to the offset of the other robot, and will push or pull until the other robot's link breaks), possibly obscuring other interesting phenomena that we might otherwise observe in the statistics. (ii) Likewise, using correction vector factors smaller than 1.0 can result in too many link failures. (iii) The rotation vector factors have positive but only slight impact on performance.

2. To examine the effect of parameter x , we choose several values a_1, \dots, a_k in the range of x , create k copies S_1, \dots, S_k of S , and change the values of x in S_i to a_i . Then for several different ladder lengths ℓ (typically between 240 and 480) and for each S_i , we run all 100 pairs of robots in the set and count how many pairs successfully carry the ladder, how many fail by getting stuck, and how many fail by breaking the linkage. We also measure the average number of steps in the corner and the average path lengths of the robots in the corner, over all success pairs.

5 Analysis of Simulation Results

We analyze the simulation results below. Due to space limitation we are not able to include actual statistics. The missing details may be found in [5].

5.1 Effect of Lookahead $lookahead_i$

Recall that larger $lookahead_i$ causes robot R_i to look farther on its guide path to choose its current target position. The simulation results indicate that, as the lookahead increases: (1) The success rate increases (until $lookahead_i$ reaches about 16). This is especially evident for longer ladders for which the failure rate is high. (2) The number of steps in the corner decreases. (3) The corner path lengths of the robots decrease significantly. The decrease is especially substantial for shorter ladders for which there is extra room to cut corners using high lookahead.

In summary, for both performance and robustness, it is important to look ahead a certain distance along the guide path to select the current target direction. For our guide path consisting of a total 160 points, the best lookahead seems to be around 16. This result may be used as a rough guide in selecting a lookahead value for similar algorithms using a guide path approach.

5.2 Effect of α_i of Guide Path

Recall that larger α_i results in a more outer guide path. The simulation results indicate that, as α_i increases: (1) The success rate increases, and the increase is especially substantial for longer ladders (e.g., for a ladder of length 480, the success rate increases from 27% to 89% when α increases from 0.40 to 0.80). The difference is less significant for shorter ladders. (2) The number of steps in the corner decreases substantially for long ladders, because ladder-wall collision is less likely and hence the robots can have much smoother motion and travel a larger distance in each step.

As α_i decreases, on the other hand, the corner path lengths decrease due to corner cutting, though for long ladders the decrease is not significant. For shorter ladders both the corner path lengths and the number of steps in the corner decrease substantially. This is because there is more room for cutting corners when the ladder is short.

In summary, larger α_i is recommended for longer ladders. Though the corner path lengths increase slightly, a much better success rate more than makes up for it. As the ladder gets shorter, we may prefer smaller α_i for a substantial reduction in both the path lengths and the number of steps.

5.3 Effect of Correction Vector Factor $k_{c,i}$

Recall that $k_{c,i}$ affects the magnitude of the correction vector \vec{c}_i that is intended to keep the offset to within the offset limit $olim_i$ and reduce the chances of linkage breaks.

The results indicate that the success rate decreases dramatically when $k_{c,i}$ decreases from 1.0 to 0.75 or 0.50 regardless of the ladder length, where as expected, almost all failures are due to linkage breaks. Therefore $k_{c,i}$ should be set to 1.00 to achieve a good success rate. Another implication of this observation is that the correction vector is working as intended when $k_{c,i} = 1.0$.

5.4 Effect of Offset Resistance Starting Point $orsp_i$

Recall that correction vector \vec{c}_i kicks in only after the offset exceeds the offset resistance starting point $orsp_i$. The results indicate that $orsp_i$ has little effect on the success rate if its value is between 0 and about 0.6 of the offset limit $olim_i$. If $orsp_i$ is 0.8 of $olim_i$ or greater, then the success rate decreases for longer ladders. The effect of $orsp_i$ on the performance is minimal, with only a slight decrease in the number of steps in corner as $orsp_i$ increases.

In summary, we can use a nonzero $orsp$ without risk of more failures as long as $(olim_i - orsp_i)$ is large enough relative to the robot's maximum speed so that the correction vector \vec{c}_i can come into play. However, the benefits of having a nonzero $orsp_i$ is limited to just a slight decrease in the number of steps in the corner.

5.5 Effect of Rotation Vector Factor $k_{r,i}$

Recall that the rotation vector factor $k_{r,i}$ controls the magnitude of the rotation vector \vec{r}_i . The results indicate that $k_{r,i}$ has very little effect on the success rate. Increasing k_r from 0 to 1.0 generally results in a slight decrease in both the average path lengths and the number of steps in the corner. In summary, using nonzero $k_{r,i}$ has positive but very light impact.

5.6 Effect of Maximum Speed $v_{max,i}$

The results indicate that as the speed of one robot decreases relative to that of the other, the success rate decreases noticeably. The decrease appears to be especially drastic for shorter ladders — for instance, when $v_{max,1}$ of R_1 is 0.5 of $v_{max,2}$ of R_2 , the success rate for a ladder of length 240 is only about as good as that for length 420. All failures are due to linkage breaks, and this is explainable: The faster robot inevitably pulls away from, or pushes toward, the slower robot stretching their offsets very close to the offset limit, leaving little room for the correction vectors \vec{c}_i to take effect.

As the difference in the maximum speeds of the robots increases, the number of steps in the corner increases greatly. Again this is expected, since if the linkages do not break, then the robots are essentially traveling at the speed of the slower robot.

Interestingly, for longer ladders (lengths 360, 420, 480), placing the faster robot up front (as R_1) results in a slightly better success rate than placing it at the back (as R_2). This seems to coincide with our intuition that when two persons carry a ladder around a corner, usually the front person should control the speed of progress, because if the back person walks too fast then the front person would be forced into the wall before he or she can turn the ladder.

In summary, in terms of robustness the algorithm is sensitive to the difference in the robots' maximum speeds.

5.7 Effect of Spring Constants sc_i

As the difference between the spring constants of the two robots increases, the success rate drops due mostly to linkage breaks. The drop is especially dramatic at longer ladder lengths. The reason for the drop is that the robot having a larger spring constant, say R_1 , has a smaller offset than the other robot, say R_2 , and thus R_1 may continue to push or pull the ladder even when R_2 's offset has reached the offset limit, making it difficult for R_2 's correction vector \vec{c}_2 to prevent a linkage break. So in terms of robustness the algorithm is sensitive to differences in the spring constants of the robots.

6 Conclusions

The detailed analysis of the results presented in Section 5 indicate that the algorithm is reasonably robust against the differences in the robots' characteristics, except their maximum speeds and spring constants. Since no two physical robots are truly identical, this robustness makes the algorithm more usable in real applications. According to the results, two physical robots executing this algorithm should be able to carry a ladder successfully and efficiently

with a high probability, if they use an outer guide path for robustness, reasonably high lookahead to improve performance by cutting corners, and a correction vector factor of 1.0 together with a sufficiently small offset resistance starting point to prevent linkage breaks.

As explained earlier, linkage breaks due to wide differences in the robots' maximum speeds and spring constants are inherently unavoidable in our model where, in a sense, each robot is allowed to "use up" its full offset range. We are currently working on alternate methods for computing the correction vectors to reduce the chances of this problem. Other issues for future research include new guide path generation methods using only local sensors, and evaluation of the algorithm's tolerance against sensor and control errors.

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," *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," *Proc. 1999 IEEE Int. Conf. on Systems, Man and Cybernetics*, Vol. VI, pp. 608–613, October 1999.
- [3] Y. Asahiro, E.C.-H. Chang, A. Mali, S. Nagafuji, I. Suzuki, M. Yamashita, "Distributed motion generation for two omni-directional robots carrying a ladder," *Proc. DARS 2000, Distributed Autonomous Robotic Systems 4*, Springer, to appear.
- [4] H. Asama, M. Sato, L. Bogoni, H. Kaetsu, A. Matsumoto, and I. Endo, "Development of an omnidirectional mobile robot with 3 DOF decoupling drive mechanism," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1925–1930, 1995.
- [5] E.C.-H. Chang, Distributed motion coordination of two robots carrying a ladder in a corridor with a 90 degree turn, MS Thesis, EECS Department, University of Wisconsin–Milwaukee, 2000.
- [6] 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*, Vol. 13, No. 5, pp. 721–729, 1997.
- [7] D. Hearn and M.P. Baker, *Computer Graphics C Version, Second Edition*, Prentice Hall, 1997.
- [8] K. Kosuge and T. Oosumi, "Decentralized control of multiple robots handling and objects," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 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] 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.