Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

Real-time Tight Coordination of Mobile Manipulators in Unknown Dynamic Environments*

John Vannoy
IMI Lab, Dept. of Computer Science
University of North Carolina - Charlotte
Charlotte, NC 28223, USA
jvannoy@uncc.edu

Jing Xiao
IMI Lab, Dept. of Computer Science
University of North Carolina - Charlotte
Charlotte, NC 28223, USA
xiao@uncc.edu

Abstract— **This paper considers the problem of planning closed-chain motion for a pair of mobile manipulators to transport a common payload in a dynamically unknown environment (i.e., an environment with moving obstacles of unknown motion). We present a novel algorithm to plan the actions of the two robots in the team, one *leader* and one *helper*, in real-time to accomplish the task while avoiding other obstacles in the unknown dynamic environment. Our algorithm does not assign fixed roles to the two team members, but rather it dynamically decides who should lead based on circumstances to optimize the team's performance. The planner is readily extensible to handle a team of more than two robots in tight collaboration. The approach is implemented and tested in simulated task environments, which demonstrate the planning algorithm's effectiveness and efficiency.**

Index Terms— **real-time, cooperative transport task, tight coordination, closed-chain, multiple mobile manipulators, adaptive, dynamic obstacles of unknown motion**

I. Introduction

In order to transport an object too heavy/large for one robot, two or more robots need to carry the object as a common payload together and transfer it. This requires tight coordination of the motions of the robot manipulators in a closed-chain system. There has been much literature addressing various aspects of tight coordination control and dynamics (e.g., [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]).

As for planning motions, an approach was introduced in [12] for three fixed-base 6-DOF manipulators to re-orient a large object together. The approach computed differential motions for the three arms numerically. No obstacle other than the robots and the manipulated object was considered. A general scheme for planning optimal trajectories and actuator forces/torques was introduced in [13] for two mobile manipulators pushing a common object to a goal location. Although obstacles were considered, they were static. In [14], a control framework decoupling locomotion (base motion) and manipulation (arm and gripper motion) was introduced for a team of two or three mobile manipulators to transport a large object. The robots were designed in such ways to enable the decoupling. The team leader could be temporarily replaced by a member to handle an obstacle.

However, no previous work has addressed the problem of planning high-DOF tight-coordination motion of mobile manipulators in an environment *with dynamic obstacles of unknown motion*. In fact, there is relatively little research on motion planning in such a dynamically unknown environment for even a single high-DOF mobile manipulator. Such an environment is very different from *known* static or dynamic environments (i.e., with known obstacle trajectories), where motion planning can reasonably rely on exploring C-space or configuration-time space *offline* (e.g., [15][16][17][18]). A few researchers considered local collision avoidance of unknown, moving obstacles on-line for a mobile manipulator base, while its arm follows certain given contour [19][20][21][22]. A more recent work can provide globally task-consistent motion in dynamic environments [23].

Recently the authors introduced a real-time, adaptive motion planning (RAMP) paradigm for a single manipulator or mobile manipulator working in a dynamic environment with unknown obstacle trajectories [24][25][26]. This paradigm is characterized by *on-line, simultaneous* planning and execution of robot motion. It borrows the general *anytime* and *parallel* planning idea of evolutionary computation [27] but is otherwise unique and original as it does not follow prescribed methods.

In this paper we address the problem of planning closed-chain motion for a pair of mobile manipulators to transport a common object in a dynamically unknown environment. We present a novel approach to plan the high-dimensional motions of the robot team in real-time with dynamic leader-helper role switch based on circumstances. This approach builds upon the RAMP paradigm ([24], [25]) to plan the motion of the leader, extends that to plan the motion of the helper, and adds a novel mechanism to handle the interaction of the two robots in the team and their flexible switch of roles. Our approach is distributed in the sense that each robot in the team has its own instance of the same planner even though they play different physical roles in their coordinated movements (i.e., one leads and one helps/follows). In section II, we further define the problem of this work. In sections

*The authors gratefully acknowledge the support of K. C. Wong Education Foundation, Hong Kong.

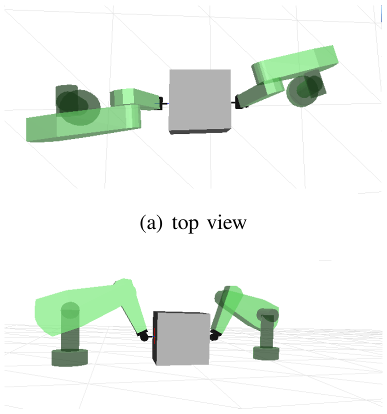
III–V, we introduce our approach. We provide implementation, experimental results, and discussions in section VI and conclude the paper in section VII.

II. Problem and Assumptions

We consider two mobile manipulators carrying a common payload A together from one location to another. We are only concerned with the coordinated transfer motion of the robots once they lift A together and do not consider grasping issues here. Therefore, we assume that (1) the end-effector grasping configurations of the two robots are known with respect to A, (2) once the end-effectors of the two robots are at the two grasping configurations respectively, the robots can lift the object together and be ready to go, (3) the two grasping configurations are assumed to be rigidly attached to A so that there is no rotation of A relative to the end-effector of any robot. Figure 1 shows two grasping configurations. Once A is carried by the two robots, it is assumed to be rigidly attached to their end-effectors, and we do not consider force control and system dynamics in this paper. A control scheme based on [9] could be extended here.

Now the problem we consider can be defined as: planning motion trajectories for two mobile manipulators that start at initial configurations where their end-effectors are rigidly attached to a common object A, have to move through an environment with obstacles of unknown motion, and transfer A to a goal location. Holonomic bases are assumed for the manipulators in this paper.

Clearly the motions of the two robots in tight coordination should be planned and executed in real-time to deal with an environment of unknown changes due to moving obstacles of unknown motions. Such coordinated motions usually require two robots play different roles: one leads the move and the other assists. It is desirable to allow switch of roles dynamically during the transport process, in order to better adapt to changes in the environment and maximize the performance of the team.


(a) top view

(b) side view

Fig. 1. Two grasping configurations for the two robots

III. Approach

The main idea of our approach is as follows. First, each mobile manipulator in the two-robot team is equipped with its own instance of the same real-time planner, which we call a *leader-oriented* planner. Each mobile manipulator uses its own leader-oriented planner to plan its motion as the *leader* and the other robot's motion as the *helper* following the leader. The leader-oriented planner, to be detailed in the following subsections, is an anytime motion planner in the RAMP paradigm introduced in our prior work [24][25][26] that integrates planning, sensing, and execution of motion to allow simultaneous motion planning and execution.

The two robots run their respective planners in parallel at the same time as they move, but the actual motions that the robots execute are determined by a simple *coordinator* algorithm running on top of the two parallel planners. The coordinator algorithm simply examines the outputs of the two motion planners constantly to decide which robot's leader-helper motion plan is better, based on some optimization criteria. It then lets the two-robot team to execute the better leader-helper motion plan, i.e., make the robot with the better plan the leader, and the other robot the helper in actual execution. As the robots move, their respective leader-oriented planners continue running to generate better motion plans for the rest of the journey from their respective perspectives. If, at some point, the current helper robot's leader-helper motion plan becomes better, the coordinator may order the team to execute this better motion plan and effectively switch the roles between the two robots. At a later moment, the robots may again switch roles by executing the better leader-helper motion plan at that time. Thus, role switching is merit-based, dynamic and adaptive to circumstances.

Note that the cost of switching from a pair of leader-helper motion trajectories to another in mid-execution is taken into account by the coordinator algorithm in making its decision. Only when switching is clearly advantageous even with its cost considered, will the coordinator algorithm order a switch of roles (and trajectories).

Note also that when there is no role-switching, the two-robot team is guided by the leader-oriented planner of the current leader robot. The leader-oriented planner will always make the team execute a better pair of leader-helper trajectories it produces at any time. That is, the team may constantly switch from executing one pair of trajectories to executing a better pair from the same planner, if such switching is advantageous with the cost of switching taken into account.

In the following sections, the leader-oriented planner and the coordinator algorithm are described in more detail.

IV. The Leader-oriented Planner

This planner plans the leader motion based on an anytime algorithm introduced in our prior work [24][25][26] and then produces a corresponding trajectory for the helper. In planning for the leader motion, it treats the payload A as a "tool" attached to the end-effector of the leader robot such that the grasping configuration of the helper end-effector becomes the configuration of the "tool" frame at the "tip" of the "tool", as shown in figure 2.

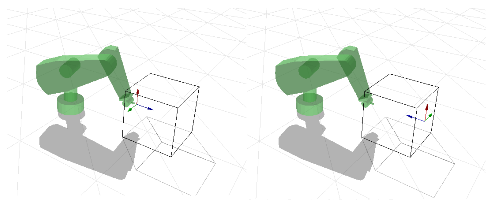


Fig. 2. Leader's end-effector frame (left) and Helper's end-effector frame (right)

Once a trajectory for the leader robot is produced, which includes both arm and base trajectories, the end-effector trajectory for the helper robot is also decided. Given the end-effector trajectory for the helper robot, the planner next finds arm and base trajectories for the helper robot to satisfy the required end-effector trajectory. We first review how to generate leader trajectories based on the RAMP paradigm and next describe how to find helper trajectories in the subsections below.

A. Planning the leader motion

A *path* of the leader robot starts from the current configuration of the leader mobile manipulator, ends at a goal configuration, and may consist of a number of intermediate configurations called *knot configurations*, specifying the shape of the path. Correspondingly, a *trajectory* of the leader robot consists of a base trajectory of the type of linear-with-parabolic blends (i.e., parabolic around knot configurations) and an arm trajectory of the type of cubic splines (connecting knot configurations). Between two adjacent knot configurations is a *trajectory segment*.

We call a trajectory of the lead robot *feasible*, if it is collision-free and singularity-free; otherwise, it is *infeasible*.

The anytime motion planning algorithm introduced in the RAMP paradigm [24][25][26] works as follows: it always maintains a set of trajectories for a mobile manipulator. A trajectory is evaluated through an *evaluation function* coding certain optimization criteria for its *fitness*. A feasible trajectory is considered fitter than an infeasible trajectory. A trajectory is generated from a path by making the robot moving as quickly as possible under a maximum speed and a maximum acceleration. The initial set of paths can be generated randomly with a random number of randomly selected knot configurations. The initial set of trajectories is then improved to a fitter set through iterations of improvements, called *generations*. In each generation, a trajectory is randomly selected from the set and altered in a random fashion by a randomly selected modification operator among a number of different modification operators, and the resulting trajectory is used to replace a similar but worse (i.e., less fit) trajectory to form a new generation. Therefore, the overall fitness of trajectories improves from generation to generation while sufficiently diverse trajectories are maintained. Each generation is also called a *planning cycle*.

This algorithm can quickly improve a set of trajectories such that the best trajectory in the set is either feasible or partially feasible while satisfying certain optimization criteria (to be explained later in Section IV-D), and it can be readily executed. The algorithm is designed to exploit the redundancy of a mobile manipulator effectively.

B. Planning the helper motion

Given a trajectory of the leader robot, the corresponding trajectory of the end-effector of the helper robot is also known. Our strategy to plan the helper motion avoids numerical computation (such as the pseudo-inverse of a Jacobian) as much as possible to achieve real-time performance. We first plan a base trajectory for the helper robot according to its end-effector trajectory. We do that by considering the constraints on the base configurations imposed by the inverse kinematic solutions of the helper robot.

If the manipulator has closed-form inverse kinematic solutions, the constraints can be found easily. For a PUMA manipulator, for example, if (b_x, b_y) indicate a base position, then (b_x, b_y) has to satisfy the following constraints in order for the end-effector to reach a given position (p_x, p_y, p_z):

$$R_1 \leq (b_x - p_x)^2 + (b_y - p_y)^2 \leq R_2 \qquad (1)$$

where

$$R_1 = d_3 \qquad (2)$$

$$R_2 = 2a_2\sqrt{a_3^2 + d_4^2} + a_2^2 + a_3^2 + d_3^2 + d_4^2 - p_z^2 \qquad (3)$$

and a_2, a_3, d_3, and d_4 are link parameters of the PUMA [28]. Clearly the above constraints define a circular belt region for allowable base positions, which we call the *allowable region* of base positions for the end-effector position (p_x, p_y, p_z), as shown in figure 3.

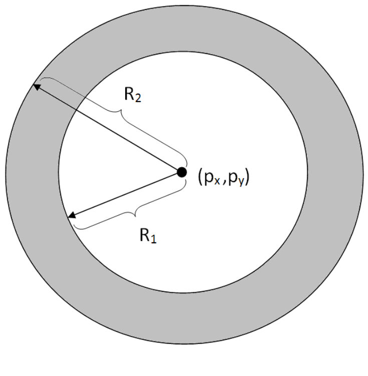


Fig. 3. Valid base region for end-effector position (p_x, p_y, p_z) given by radii R_1 and R_2

We plan the trajectory of the helper base in the following way. For each knot configuration on the trajectory of the leader robot, we get the corresponding end-effector configuration of the helper robot (or the tool frame of the leader robot), and from the allowable region of the helper base configurations, we randomly pick a knot configuration for the base. In this way, we get a path of the helper base as a sequence of knot configurations corresponding to the sequence of knot configurations of the leader base. Next we make a smooth base trajectory of linear-with-parabolic blends for the helper based on its knot base configurations in the same way as the leader's base trajectory is made.

The arm trajectory of the helper corresponding to the base trajectory and the end-effector trajectory can be generated by discretizing the base and the end-effector trajectories into a pair of discrete time series, and by finding the corresponding time series for joint variables through inverse kinematics at each discrete time. Closed-form inverse kinematic solutions make the computation very fast.

A trajectory of the helper robot is *feasible* if it is (1) collision-free, (2) singularity-free, and in addition, (3) satisfies the end-effector trajectory determined by the leader robot.

Condition (3) may be violated in two cases. One is that the generated helper base trajectory may not be in exact synchronization with the corresponding leader's base trajectory or consequently the helper end-effector trajectory. This is because of the redundancy in the form of one-to-many mapping between an end-effector location and a base location. As the result, there can be a mismatch between a base location and the end-effector location at a certain time so that no inverse kinematic solution exists for the arm. The other case is that the base path itself may include points outside the allowable region of the base locations, which is the union of the allowable regions for each end-effector location along the path of the end-effector. Since this region is generally not convex, even though we make sure that the knot configurations of the base path are in the region, there is no guarantee that the entire path is inside the allowable region.

We describe how to deal with infeasible trajectories in the following subsections.

Note that if a trajectory of the helper robot is feasible, it is also smooth: both the end-effector and the base trajectories are smooth, and so the arm trajectory generated also converges to a smooth one as the time interval approaches zero for the time series.

C. Trajectory feasibility and improvement as a team

As described above, from a trajectory of the leader, a corresponding trajectory of the helper can be generated. A pair of leader and helper trajectories form a *team trajectory*. A team trajectory is infeasible if either the leader or the helper trajectory is infeasible.

One may ask that if a leader trajectory is infeasible, why the planner bothers to generate the corresponding helper trajectory anyway. This in fact is a characteristic of the leader-oriented planner carried over from the anytime planner of the RAMP paradigm. In planning the leader motion, the anytime planner always maintains a set of leader trajectories, which may or may not be all feasible. Currently infeasible trajectories may become feasible in some future moments under the changing environment, whereas currently feasible trajectories may become infeasible. Moreover, some infeasible trajectories may easily be improved to be good feasible ones after certain modification operators are applied (see section IV-A). Therefore it is useful to keep infeasible trajectories in the set of leader trajectories, and consequently, the corresponding helper trajectories are generated.

The leader-oriented planner makes sure that right after a leader trajectory is created, a corresponding helper trajectory is also created, and right after a leader trajectory is modified, the corresponding helper trajectory is also modified. In that sense the leader-oriented planner maintains a set of team trajectories and improves/adapts them from one planning cycle to the next as described in section IV-A.

In addition to modifying a leader trajectory by one of a number of randomized modification operators (section IV-A) and then adjusting its corresponding helper trajectory accordingly, the leader-oriented planner also introduces a new randomized modification operator **perturbHelper** to modify a base trajectory of the helper only. Therefore, if a leader trajectory is a good and feasible one, but the helper trajectory is not feasible (see section IV-B), by allowing the base trajectory of the helper to change, the helper trajectory may become feasible without disturbing the leader trajectory or the end-effector trajectory of the helper. The redundancy of the mobile manipulator can be effectively exploited.

D. Fitness evaluation

We use two different cost functions to evaluate the fitness of feasible and infeasible trajectories respectively. The higher the cost, the less fit a trajectory is. For each feasible trajectory we compute its fitness value through a cost function that combines three optimization criteria: minimizing energy and time, and maximizing manipulability [26][29].

For each infeasible trajectory, we compute its cost as the sum of two terms. The first term is the cost as if it were feasible, and the second term is a large penalty function value that is inversely proportional to the time before the first infeasible configuration on the trajectory is encountered: the shorter time, the bigger cost.

With these cost functions, a feasible trajectory is always fitter than an infeasible trajectory.

The above functions apply to both leader and helper trajectories. To evaluate a team trajectory, we simply use the sum of the costs of the leader and helper trajectories in the team. Note that for a feasible team trajectory, we can also just use the cost of the leader trajectory alone since the helper trajectory is closely related to the leader trajectory.

V. The Coordinator and Real-time Adaptiveness

Recall that each robot in a two-robot team runs its own instance of the leader-oriented planner from its own perspective, i.e., itself is considered the leader robot and the other team member is viewed as the helper robot. Which robot's leader-helper motion plan is actually executed by the team is decided by a small coordinator algorithm.

The coordinator algorithm can actually reside on the processor of one of the robots but it treats both robots impartially. It compares the best team trajectory generated by robot 1's leader-oriented planner with the best team trajectory generated by the robot 2's leader-oriented planner, based on their fitness evaluation function values. If robot 1's team trajectory is better, the coordinator will let the two-robot team execute that trajectory and make robot 1 the actual leader. If robot 2 is the current leader, executing robot 1's team trajectory means a switch of roles between robot 1 and robot 2.

One basic premise of our approach is that planning, sensing, and the execution of motion are interweaving to enable simultaneous planning and execution of robot motions. After some initial planning cycles, the coordinator can make an initial leader-helper assignment based on the planning outputs of the leader-oriented planners of both robots. The robot team can then start executing the chosen team trajectory from the planner of the leader robot in the first control cycle. As the robot team moves, each robot's planner continues to improve its set of team trajectories until the next control cycle[1], when the robot team can switch to a better team trajectory (recall that the set of trajectories always start from the current configurations and velocities of the robot team). One of two kinds of switching can happen: (1) the coordinator can make the robot team switch to a team trajectory from the planner of the current helper robot, i.e., make the robots switch roles, or (2) the robot team can switch to a better team trajectory from the planner of the leader robot, and the current leader robot continues to lead. In both cases, the new team trajectory is indeed better even after taking into account the cost of change (i.e., the possible acceleration or deceleration needed for the change) as ensured by the fitness evaluation functions so that the change is smooth and stable.

Note that the best team trajectory does not have to be feasible; if no feasible trajectory is available, the robot team will move along the fittest infeasible trajectory while continuing planning to search for a fitter and hopefully feasible team trajectory before the team come within a distance threshold D of the first predicted infeasible configuration on the executed trajectory. In the event D is reached but no fitter team trajectory is available, the two robots will stop their motions but continue planning for a fitter team trajectory and resume their motions once a better team trajectory is found. Such stops of motion are called *forced stops*.

Changes in a dynamic environment are sensed and fed to the planner in each sensing cycle, which lead to updated fitness values for certain team trajectories in the subsequent planning cycles of either robot's planner. Unknown motions of moving obstacles are predicted in fitness evaluation of trajectories. Our planner predicts the future trajectory of each moving obstacle body from its current sensed state (i.e., configuration, velocity, and acceleration) and previously sensed trajectory and checks each robot's trajectory against this predicted or projected trajectory of each obstacle to see if there will be a collision. Our prediction only has to be good enough for a short period before the next sensing cycle (which may be longer or shorter than a control cycle) since it will be corrected constantly with newly added sensory information.

[1]A control cycle in this sense is longer than a planning cycle to use the planning results as feedback. It is not necessarily the lowest-level servo cycle.

Note that when evaluating a trajectory of a helper robot, the leader robot (its base and links) and the common payload should be considered dynamic obstacles.

The presence of a diverse set of ever-improving trajectories enables the robot team to quickly adapt to changes in the environment by following the fittest team trajectory under each circumstance: when the current trajectory executed by the team becomes worse, the robots often do not need to stop and replan from scratch; rather they merely need to switch to a better team trajectory in the set swiftly in a seamless fashion. The chosen team trajectory can be very different from the previous one (with even a change of the leader robot) to deal with drastic and large changes. With both robots running their own planners in parallel, the pool of team trajectories (though from different perspectives) doubles in size, and so does the planning cycle frequency, to facilitate highly efficient and effective task performance.

VI. Implementation, Results, and Discussion

In this section we present our implementation results and discuss the performance of our approach.

A. Implementation

In order to test the introduced motion planner, we build a mobile manipulator simulator for a PUMA 560 mounted to a holonomic mobile base. We use several such mobile manipulators in our experiments. Both the mobile manipulators and the objects in the environment are modeled as polygonal meshes for generality. We use the software package OPCODE [30] to perform real-time collision detection for feasibility evaluation of a robot trajectory.

Each mobile manipulator is equipped with its own instance of the leader-oriented planning algorithm, which has no a priori knowledge of the movements of other mobile manipulators and moving obstacles outside the team. Each mobile manipulator views another mobile manipulator as consisting of 7 or 8 moving bodies (as obstacles) due to the number of links (including load) of each mobile manipulator, with the number of bodies depending on if the other mobile manipulator holds a target object or not. We implemented the planning approach (including the leader-oriented planning algorithm and the coordinator) in C# and C++, and have simulated task environments with 1–2 pairs of mobile manipulators. Each task simulation is run on a four-core Xeon PC with each core operating at 3.0 GHz.

In our experiments, we set the following parameter values. The weight of the manipulator arm and the base are set to be 35 kg and 20 kg respectively. The maximum joint velocity and acceleration for the PUMA are set to be 120 deg/sec and 60 deg/sec^2 respectively. The maximum base velocity and acceleration are set to be 2 m/sec and 1 m/sec^2 respectively. The work environment is a square of flat area with the side length 100 meters. The frequency of the control cycle for a mobile manipulator is set to be 60Hz. The control cycle is therefore quite slow, as compared to the planning cycle, which has a frequency many times that of the control cycle, depending on the task environment.

B. Performance Evaluation

Figure 4 shows a task environment we use for performance evaluation of our real-time motion planner for tight coordination. In that task environment, there are four boxes on the floor, and two pairs of mobile manipulators are to transport four boxes from their original locations to designated goal locations. Each box has to be carried by a pair of mobile manipulators. The environment is arranged such that two teams of robots must cross paths and therefore must avoid each other in their movement. This is the case both when the robot teams are transporting boxes, and also when the robots are moving individually on their way to reach a box. Note that in the case multiple mobile manipulators have to move individually to reach boxes, each robot uses the basic RAMP algorithm for a single mobile manipulator to plan and execute its motion in real-time while avoiding other robots and obstacles [29].

We compare the effect of dynamic role switching to that of static role assignment for both teams of robots. A team's performance is measured first with static roles (i.e. the leader and helper roles are pre-decided and static), and again with dynamic role switching. The results are summarized in table I.

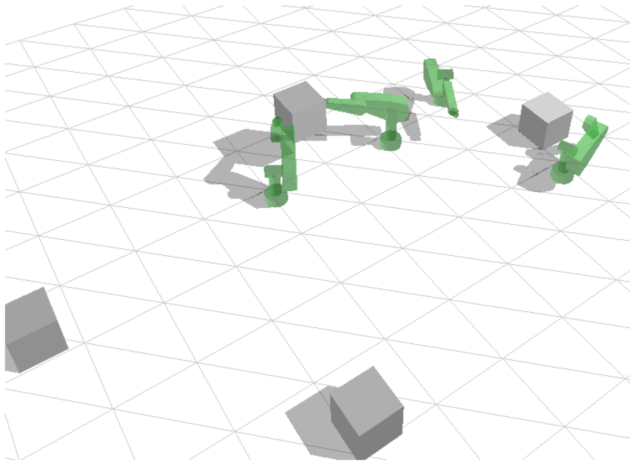


Fig. 4. Task environment

TABLE I
TASK EXECUTION WITH AND WITHOUT DYNAMIC ROLE SWITCHING
(AVERAGE OF 25 EXECUTIONS)

(a) Team 1

Role Switching	*# Role Switches*	*Total Time(s)*	*# Planning Cycles*	*Plan per Control*	*# Forced Stops*
with	181	116.7	15,709	2.24	4
without	n/a	128.8	17,907	2.32	5

(b) Team 2

Role Switching	*# Role Switches*	*Total Time(s)*	*# Planning Cycles*	*Plan per Control*	*# Forced Stops*
with	153	106.4	12,961	2.03	2
without	n/a	121.0	15,842	2.18	4

For both teams, with dynamic role switching, we see a reduction (9.4% and 12.1%) in total execution time respectively, and also a reduction in the number of times a team is forced to stop motion to avoid a collision during the execution. The execution time is measured from the time when a pair of robots lift a box to the time when the team puts the box down at a destination. In the cases without dynamic role switching, only the leader performs motion planning and therefore the values shown under "# Planning Cycles" and "Plan per Control" (i.e., planning cycles per control cycle) are solely the leader's. In the cases with dynamic role switching, both the leader and helper contribute to the motion planning, and those values are the average of the two robots. Because of the small overhead cost of the coordinator algorithm to switch roles, these values are slightly lower than those of the static-role cases. Nevertheless, each team's performance is improved with dynamic role switching due to having two different robots perform planning actively from two different perspectives.

Allowing both members of a robot team to perform planning is advantageous for two reasons. First, it virtually doubles the effective planning cycle frequency, and also doubles the effective size of the trajectory pool, since both robots are planning toward the same goal state. Second, it permits planning from the different perspectives of the two team members. This is important because the arrangement or motion of obstacles in the environment might make one planning perspective more advantageous than another.

Note that the number of forced stops is fairly low in all cases, indicating that our approach is able to provide good motion plans in real time most of the time.

The accompanying video shows task executions in two task environments. In the first environment, two robot teams perform the payload transport task described above while avoiding each other and other obstacles. In the second environment, one robot team transports and reorients a flat table, while avoiding an obstacle of unknown movement; the dynamic role switches during the process are also indicated in the second environment, which is deliberately kept simple to show the fine maneuvering clearly.

C. Scalability

The above approach is not limited to a team of two mobile manipulators. When two or more helpers are involved, the leader-oriented planner just needs to generate two or more helper robots' motion instead of one, and each robot, again has its own instance of the planner. By having all robots running their respective leader-oriented planners in parallel, planning efficiency is n times of that running a single leader-oriented planner, if n is the number of robots in a team. This balances the fact each leader-oriented planner needs to generate $n-1$ number of helper trajectories instead of one (of a two-robot team). Moreover, n different planning perspectives are provided by the n team members.

VII. Conclusions

This paper has introduced a novel approach to real-time, distributed motion planning for a team of mobile manipulators in tight-coordination to transport a common payload in unknown dynamic environments (i.e., environments with obstacles of unknown motion). It effectively extends an efficient real-time adaptive motion planning approach for single mobile manipulators in such an environment.

The approach allows all team members to participate in planning the team motions in parallel from their respective perspectives. Each team member is equipped with its own instance of the same leader-oriented motion planner to plan the team motion from its own perspective, i.e., viewing itself as the leader and other team members as helpers. All the instances of the leader-oriented motion planner are run in parallel as the team moves. Planning and execution of motion are done simultaneously. A simple coordinator algorithm always makes the team following the best plan so far and the robot whose plan is being executed is the current leader. This means dynamic, merit-based switch of leaders according to circumstances.

With parallel, multi-perspective planning guided by optimization criteria, our approach is able to achieve both high motion quality and real-time planning efficiency for high-dimensional robots in tight-coordination motion amid moving obstacles of unknown trajectories. Future work includes further testing and improving the algorithm for more complex robots and tasks and incorporating realistic sensing scenarios and control constraints. Testing on real robots will also be necessary.

References

[1] Y. F. Zheng and J. Y. S. Luh, "Joint torques for control of two coordinated moving robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 1986, pp. 1375–1380.

[2] S. Hayati, "Hybrid position/force control of multi-arm cooperating robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 1986, pp. 82–89.

[3] T.-J. Tarn, A. K. Bejczy, and X. Yun, "Design of dynamic control of two cooperating robot arms: Closed chain formulation," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 1987, pp. 7–13.

[4] M. Uchiyama and P. Dauchez, "A symmetric hybrid position/force control scheme for the coordination of two robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 1988, pp. 350–356.

[5] O. Khatib, "Object manipulation in a multi-effector robot system," in *Robotics Research*, R. Bolles and B. Roth, Eds. MIT Press, 1988, vol. 4, pp. 137–144.

[6] D. Williams and O. Khatib, "The virtual linkage: A model for internal forces in multi-grasp manipulation," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, May 1993, pp. 1025–1030.

[7] J. Adams, R. Bajcsy, J. Kosecka, V. Kumar, R. Mandelbaum, M. Mintz, R. Paul, C. Wang, Y. Yamamoto, , and X. Yun, "Cooperative material handling by human and robotic agents: Module development and system synthesis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995, pp. 200–205.

[8] K.-S. Chang, R. Holmberg, and O. Khatib, "The augmented object model: Cooperative manipulation and parallel mechanism dynamics," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 2000, pp. 470–475.

[9] D. Sun and J. K. Mills, "Adaptive synchronized control for coordination of two robot manipulators," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, May 2002, pp. 976–981.

[10] Y. Hirata, Y. Kume, T. Sawada, Z.-D. Wang, and K. Kosuge, "Handling of an object by multiple mobile manipulators in coordination based on caster-like dynamics," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, 2004, pp. 807–812.

[11] A. Stroupe, T. Huntsberger, A. Okon, H. Aghazarian, and M. Robinson, "Behavior-based multi-robot collaboration for autonomous construction tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, August 2005, pp. 1495–1500.

[12] C. S. Tzafestas, P. Prokopiou, and S. G. Tzafestas, "Path planning and control of a cooperative three-robot system manipulating large objects," *Journal of Intelligent and Robotic Systems*, vol. 22, pp. 99–116, 1998.

[13] J. P. Desai and V. Kumar, "Nonholonomic motion planning for multiple mobile manipulators," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 1997.

[14] T. Sugar and V. Kumar, "Control of cooperating mobile manipulators," *IEEE Trans. Robotics and Automation*, vol. 18, no. 1, pp. 94–103, August 2002.

[15] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, 1996.

[16] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[17] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[18] J. van den Berg and M. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Trans. Robotics*, vol. 21, no. 5, pp. 885–897, October 2005.

[19] O. Brock, O. Khatib, and S. Viji, "Task-consistent obstacle avoidance and motion behavior for mobile manipulation," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, May 2002, pp. 388–393.

[20] P. Ögren, N. Egerstedt, and X. Hu, "Reactive mobile manipulation using dynamic trajectory tracking," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, April 2000, pp. 3473–3478.

[21] J. Tan and N. Xi, "Unified model approach for planning and control of mobile manipulators," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, 2001, pp. 3145–3152.

[22] J. Mbede, S. Ma, Y. Toure, V. Graefe, and L. Zhang, "Robust neurofuzzy navigation of mobile manipulator among dynamic obstacles," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 5, May 2004, pp. 5051–5057.

[23] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments," in *Proceedings of Robotics: Science and Systems*, Philadelphia, PA, USA, August 2006.

[24] J. Vannoy and J. Xiao, "Real-time adaptive and trajectory-optimized manipulator motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, September 2004, pp. 497–502.

[25] ——, "Real-time adaptive mobile manipulator motion planning," in *Video Proceedings of IROS*, October 2006.

[26] ——, "Real-time planning of mobile manipulation in dynamic environments of unknown changes," in *Proceedings of RSS 2006 Workshop: Manipulation for Human Environments*, August 2006.

[27] P. P. Bonissone, R. Subbu, N. Eklund, and T. R. Kiehl, "Evolutionary algorithms + domain knowledge = real-world evolutionary computation," *IEEE Trans. Evolutionary Computation*, vol. 10, no. 3, pp. 256–280, April 2006.

[28] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans ASME J. Appl. Mech*, vol. 23, pp. 215–221, 1955.

[29] J. Vannoy and J. Xiao, "Real-time motion planning of multiple mobile manipulators with a common task objective in shared work environments," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 2007.

[30] P. Terdiman, "http://www.codercorner.com/opcode.htm."