2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Real-time Motion Planning of Multiple Mobile Manipulators with a Common Task Objective in Shared Work Environments

John Vannoy
IMI Lab, Dept. of Computer Science
University of North Carolina - Charlotte
Charlotte, NC 28223, USA
jmvannoy@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 motions for a team of mobile manipulators working in the same environment with a common task objective. It presents a distributed, real-time algorithm to plan motion trajectory for each team member that allows dynamic and spontaneous division of work among team members to meet the common task objective. A mobile manipulator has to perform its share of the task while avoiding other moving mobile manipulators in the team in addition to other obstacles in the environment. To each robot team member, none of the trajectories of the other team members or moving obstacles are known beforehand. The approach is implemented and tested in simulated task environments, which demonstrates its high effectiveness and efficiency.**

I. Introduction

Our research in this paper is inspired by the scenario where a team of workers simultaneously perform the same kind of tasks in a shared environment with the common objective being to get the job done as quickly as possible. Examples include moving chairs from a storage place to a ballroom and arrange them for a meeting (or conference keynote presentation, etc.), loading/unloading goods for a store (or for disaster relief, etc.), picking up cans, bottles, and other garbage in an open space after a public gathering, and so on. Each worker makes his/her individual movement but shares the set of target objects with other workers so that the workers partition the target objects. However, the partition is usually not done in advance but is dynamically and spontaneously created as the workers make their movements to perform the task. This usually results in an efficient and flexible division of work.

For a group of mobile manipulators to perform such kind of tasks, they should also dynamically and spontaneously divide their targets with each individually deciding its own movements. Therefore, offline and centralized motion planning methods are not suitable. We are interested in real-time, distributed motion planning here to enable each mobile manipulator to decide its own motion and targets based on circumstances while avoiding other mobile manipulators and moving obstacles, whose trajectories are unknown beforehand. This is largely an open problem. The existing literature has only addressed some related aspects.

There is much research on task allocation for multiple mobile robots, which is about how to divide a task into subtasks and assign robots to the subtasks. Existing strategies are mostly based on heuristics, including behavior-based approaches (e.g., [1], [2]), auction-based approaches (e.g., [3], [4]), dynamic token passing [5], etc. Such research often aims at tasks that either require more sophisticated coordinations among robot team members, e.g., cooperative pushing, or require different kinds of subtasks or different roles for different robots. The focus is often not on planning actual physical motions of the robots, and only (low-dimensional) mobile robots are assumed. In [6], a task of collecting pucks in the environment is divided among a group of mobile robots through assigning each robot its own spatial territory so that no two robots share the same workspace.

There is relatively little research on real-time motion planning for a high-dimensional robot, such as a manipulator or a mobile manipulator, in a dynamic environment with *unknown* obstacle trajectories. 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., [7][8][9][10]). 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 [11][12][13][14]. A more recent work can provide globally task-consistent motion in dynamic environments [15].

Recently the authors introduced a real-time, adaptive motion planning paradigm for a single manipulator or mobile manipulator working in dynamic environments with unknown obstacle trajectories [16][17][18]. 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 [19] but is otherwise unique and original as it does not follow prescribed methods.

In this paper we present a novel approach for distributed, real-time planning of motions for a team of mobile ma-

nipulators who work in the same environment with a common task objective and who dynamically and spontaneously divide the work. The approach extends our prior work in [16][17] for a single mobile manipulator. In section II, we formally define the problem of this work. In section III, we introduce our approach. We provide implementation, experimental results, and discussions in Section IV and conclude the paper in Section V.

II. Problem

We can define the motion planning problem of our concern in this paper as the following. A team of M mobile manipulators need to pick up N objects ($M < N$) and move them to some destinations. Each robot can only pick up and move one object at a time. The initial and goal locations of the ith object is denoted as $C_{i,i}$ and $C_{i,g}$ respectively, where $C_{i,g}$ may need to be decided dynamically. The set of objects that the mobile manipulator j picks up and moves is denoted as S_j, with cardinality n_j. S_j is determined dynamically and $\Sigma_{j=1}^{M} n_j = N$. No object can be picked up by two mobile manipulators, i.e., $S_j \cap S_k = \emptyset$, for $i \neq k$. Each mobile manipulator has to avoid the motions of the rest of the mobile manipulators in the team as well as other (static or moving) obstacles in the environment while performing its task. The target objects themselves are also obstacles to the mobile manipulators before they are picked up.

From the point of view of each mobile manipulator, the working environment is dynamic because (a) other mobile manipulators are dynamic obstacles regardless if there are additional dynamic obstacles, and (b) the target objects can dynamically disappear from their initial locations (once they are picked up by other mobile manipulators) and become dynamic obstacles as they are carried by other mobile manipulators to their destinations. The motion of a mobile manipulator regarding one target object can be considered as consisting of two parts: (1) moving to pick up the target, and (2) moving while carrying the target and placing the target at a destination. Each mobile manipulator in the team will repeatedly try to pick up and move a target until all N objects are picked up or moved.

III. Approach

We present a real-time, distributed approach to planning motions of mobile manipulators in a team. There is no centralized planning. The real-time motion planning algorithm treats the task environment from the point of view of an individual mobile manipulator in a team, and the same algorithm applies to every team member robot. The algorithm treats other mobile manipulators in the team as moving obstacles to avoid. The algorithm dynamically decides, for the mobile manipulator it controls, which target object should be picked up each time the mobile manipulator is available for that, and where to move the target object. Details of the algorithm are given below.

A. Task Motion Representation

We consider each mobile manipulator repeatedly performing the task of picking up a target object and moving it to a destination until there is no target object available for picking up. The task can be divided into two subtasks: *pick-up* for the motion of going to a target object and picking it up and *put-down* for the motion of carrying the object to a destination and putting it down. Our planner first plans the motion for the pick-up subtask for a mobile manipulator in real-time, and only when the pick-up subtask is achieved, i.e., after the mobile manipulator picks up a target object, it starts planning the motion for the put-down subtask in real-time.

A *pick-up path* for a mobile manipulator starts from the current location of the mobile manipulator and ends at a configuration where the target object can be picked up, called a *pick-up configuration*. A *put-down path* starts from the current location of the mobile manipulator, which initially is at a pick-up configuration and ends at a destination configuration where the target object can be put down, called a *put-down configuration*. Either path can have a number of intermediate configurations called *knot configurations*, specifying the shape of the path.

A trajectory corresponding to a mobile manipulator path (i.e., either a pick-up path or a put-down path) consists of a base trajectory of the type of linear-with-parabolic blends and an arm trajectory of the type of cubic splines. Between two adjacent knot configurations is a *trajectory segment*.

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

If the target object for a pick-up configuration is available (i.e. has not been picked up by another robot), the pick-up configuration is called *valid*; otherwise, the pick-up configuration is *invalid*. Similarly, if the target object can be placed at a put-down configuration (i.e. the spot chosen to place the object), the put-down configuration is called *valid*; otherwise it is *invalid*.

B. Basic Planning Paradigm

One basic premise of our approach is that planning, sensing, and the execution of motion are interweaving to enable simultaneous robot motion planning and execution. This is achieved through our anytime planning algorithm that 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. The initial set of trajectories can be generated randomly with a randomly selected available target object and a random number of randomly selected knot configurations for a trajectory. The initial set 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 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*.

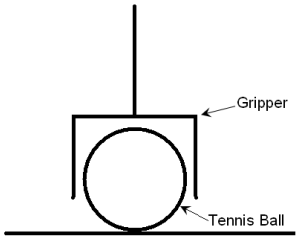
The robot can start following the fittest trajectory at the beginning of a control cycle. As the robot moves, planning continues to improve the set of trajectories until the next control cycle, when the robot can switch to a fitter trajectory so that it always follows the best trajectory. For that purpose each trajectory is always updated to start from the current robot configuration, and such updating is done once in each control cycle. Note that the fittest trajectory does not have to be feasible; if no feasible trajectory is available, the robot will move along the fittest infeasible trajectory (see section III-E for fitness measure) while continuing planning to search for a fitter and hopefully feasible trajectory before it comes within a distance threshold D of the first predicted collision or singularity of the executed trajectory. In the event D is reached but no fitter trajectory is available, the robot will stop its motion but continue planning for a fitter trajectory and resume its motion once a better trajectory is found.

Changes in a dynamic environment are sensed and fed to the planner in each sensing cycle, which lead to updated fitness values for certain trajectories in the subsequent planning cycles, and 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 a 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.

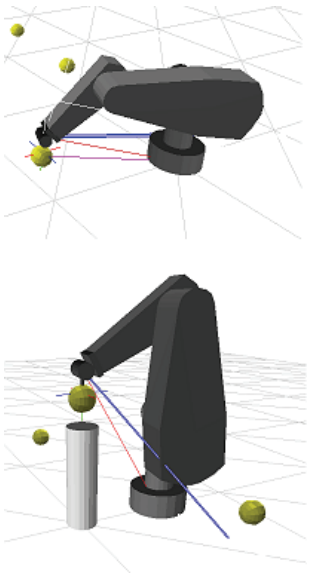
The presence of a diverse set of ever-improving trajectories enables a mobile manipulator to quickly adapt to changes in the environment by following the fittest trajectory under each circumstance: when the current trajectory that the robot follows becomes worse, the robot does not need to stop and replan from scratch; rather the planner often merely needs to switch the robot to a better trajectory in the set swiftly in a seamless fashion. The chosen trajectory can be of a very different homotopic group from the previous one to deal with drastic and large changes.

It should be emphasized that during the simultaneous planning, sensing, and motion execution, when the mobile manipulator changes course from one trajectory to another, the new 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 function so that the change is smooth and stable, and the actual trajectory executed by the robot is the best possible result.

Our approach also supports the partial specification of goals: only the end-effector position and orientation for picking up a target object or placing it down are needed. Figure 1 illustrates example pick-up and put-down configurations for the end-effector. As the result, there can be different base and arm configurations for the same end-effector goal in different trajectories so that redundancy is exploited to achieve flexibility amid environments with dynamic changes.



(a) Pick-up and put-down end-effector configuration



(b) Example pick-up configuration (top) and put-down configuration (bottom)

Fig. 1. Pick-up and put-down end-effector configuration

C. Trajectory Modification

We use the following six random modification operations to modify a trajectory with a valid pick-up or put-down configuration:

- **Insert** - a new, random knot configuration is inserted between two randomly chosen adjacent knot configurations of a path.
- **Delete** - a randomly selected intermediate knot configuration is deleted from a path.
- **Change** - a randomly selected intermediate knot configuration is replaced with a new, randomly generated knot configuration.
- **Swap** - two randomly selected adjacent intermediate knot configurations from a single path are swapped.
- **Crossover** - the knot point lists of two paths are divided randomly into two parts respectively and recombined: the first part of the first path with the second part of

the second path, and the first part of the second path with the second part of the first path.

- **Stop** - the base movement or any joint movement of the arm stops at a randomly chosen knot configuration. The duration of the stop is determined randomly.

The first five operations are used to change the shape of a path and subsequently the corresponding trajectory. The **Stop** operation is used to change a trajectory only. Our algorithm simply randomly selects one of those operations (also called operators) to apply to the selected trajectories. All operators are used to change the trajectories of the base and the manipulator either separately or together in a stochastic fashion.

The **Stop** operator enables loose-coupling between the trajectories of the base and the manipulator. Both can stop their movements independently or together. The probabilistic nature of our approach simply offers a stop as a possibility; in the cases where stopping is advantageous, the planner will utilize it.

Note that except for **Crossover**, the other operations above are unary transformations that change a single trajectory. The crossover generates two offsprings from two parent trajectories. Depending on if the selected operation is unary or crossover, one or two trajectories from the current population are selected at random. One or two new trajectories are generated by applying the selected operation to the selected trajectory or trajectories and are then evaluated.

D. Dynamic and Spontaneous Division of Work

In each initial pick-up trajectory for a mobile manipulator, an available target object can be selected randomly, and a corresponding pick-up configuration for the robot is selected accordingly. Different pick-up trajectories for the same mobile manipulator may be associated with the same or different target objects. Once the pick-up configuration becomes invalid for a pick-up trajectory (i.e., the target object is no longer available), a **Repair** operator randomly selects another available target object and a corresponding pick-up configuration to replace the old pick-up configuration in the trajectory. The **Repair** operator performs such repair of trajectories once in every sensing cycle.

Since different trajectories can be associated with different target objects, during the course of the mobile manipulator motion, it may change target objects with two reasons: (1) it may change to a better trajectory (see Section III-B) to be more efficient or to avoid the motions of the other mobile manipulators, and (2) it may follow a repaired trajectory due to the actions of other mobile manipulators. Thus, the actual target object picked-up each time by each mobile manipulator is dynamically determined. As the result, the division of N target objects among M mobile manipulators is also dynamically and spontaneously achieved.

The put-down location for a target object can be either random within a certain range (e.g., as long as the target object is placed in a basket, the goal is achieved) or has to follow certain arrangement (e.g., chairs have to be put surrounding a table). A **Place** operator is used to implement these kinds of requirements depending on specific tasks. The **Place** operator decides a put-down configuration for each initial put-down trajectory of a mobile manipulator. Different put-down trajectories for the same mobile manipulator may be associated with the same or different put-down configurations. Once the put-down configuration becomes invalid for a put-down trajectory (i.e., the intended spot for the target object is occupied), the **Place** operator decides another suitable put-down configuration to replace the old put-down configuration in the trajectory. The **Place** operator performs once in every sensing cycle.

Our planner for each mobile manipulator is designed to optimize its own motion based on certain optimization criteria (described in the following subsection). Together the M mobile manipulators will accomplish the task of picking up and moving N objects in a truly distributed, dynamic, and spontaneous fashion. This very much resembles how a team of human workers will handle the same task.

E. Fitness Evaluation

In our planner, the fitness evaluation has two components: *feasibility checking* and *optimization criteria*. We use two different evaluation functions for feasible and infeasible trajectories. In each case, the evaluation function is a cost function to measure the fitness of a trajectory. The higher the value of the evaluation function, the worse or less fit a trajectory is.

For each feasible trajectory we compute its fitness value through a fitness function that combines three optimization criteria: minimizing energy and time, and maximizing manipulability. To measure time optimality, we compute the minimum time needed for the robot to move through all path segments in cubic trajectory for the arm and in linear-with-parabolic-blend trajectory for the base, taking into account constraints on speed and acceleration of each link (including the base).

Since the motions of the mobile base and the manipulator arm are not decoupled so that they can happen together, minimum time as a measure alone cannot differentiate an efficient trajectory with minimum motion from one with unnecessary arm or base movement while still maintaining the same overall minimum time. Therefore, we use minimum energy as another measure of optimality to distinguish between two trajectories whose time requirements are equal but their energy requirements are not; the one that requires less energy is preferred.

To evaluate the manipulability associated with a collision-free trajectory, we use the average of the inverse value of the manipulability measure at each configuration [20] of the whole trajectory.

For each infeasible trajectory, we define a fitness value as the sum of a large penalty and its fitness function value as if it were feasible[1]. The large penalty term serves two purposes. One purpose is to make sure that infeasible trajectories are

[1]For a trajectory with singularities, we compute its "manipulability as if it were feasible" by excluding the singular configurations.

less fit than feasible trajectories. The other is to serve as a measure of relative safety so that infeasible trajectories with smaller penalty terms are considered safer and therefore fitter than ones with larger penalty terms. For the latter purpose, we define the penalty term of an infeasible trajectory as $\frac{P}{T_{coll}}$, where P is a large constant and T_{coll} is the time before either the first predicted collision or the first singular configuration, whichever comes first, in the trajectory. That is, we consider an infeasible trajectory safer if it has a longer time before the first predicted collision/singularity.

By allowing infeasible trajectories in the set of althernative trajectories for trajectory improvement and execution, our algorithm aggressively maximizes the chances to optimize a robot's real-time actions efficiently.

It should be noted that in addition to the above criteria, other criteria (e.g., safety and stability measures [21]) could be used and aggregated into the evaluation function, requiring changes only in the evaluation procedure, and not to the overall algorithm. Note also that regardless of whether a trajectory is feasible or infeasible, the corresponding evaluation function is computed as the sum of the costs for individual trajectory segments. This property greatly facilitates efficient evaluation of trajectories in each generation of the planning algorithm since only the altered and affected trajectory segments need to be re-evaluated, especially in real-time. The evaluation of infeasible trajectories is further speeded up by that once the first collision is detected between a single link of a robot and a single obstacle body, the entire trajectory is labeled infeasible, and no further collision checking is required.

IV. 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 mobile base. We use a team of 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 [22] to perform real-time collision detection for feasibility evaluation of a robot trajectory. Collision checking is performed at discrete points along the trajectory, at high resolution; more sophisticated approaches (e.g. [23]) could also be used here.

Each mobile manipulator is equipped with its own instance of the same real-time adaptive motion planning algorithm, which has no a priori knowledge of the movements of the other mobile manipulators and moving obstacles. 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 a mobile manipulator, with the number of bodies depending on if the other mobile manipulator holds a target object or not. Therefore, in a task environment of 3 mobile manipulators, for example, each mobile manipulator considers the other two mobile manipulators as 14–16 moving bodies, in addition to other moving obstacles that may exist in the environment. We implemented the planning algorithm in C# and C++, and have simulated task environments with 2–4 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

To test the performance of our real-time, distributed motion planner, we compare the effects of different mobile manipulator team sizes and the effects of how the pick-up targets are decided (pre-decided vs. dynamically and spontaneously decided). The task environment we use consists of 12 randomly scattered tennis balls on the floor, as shown in Figure 2. We consider mobile manipulator teams of different sizes performing two different tasks with varying levels of complexity.

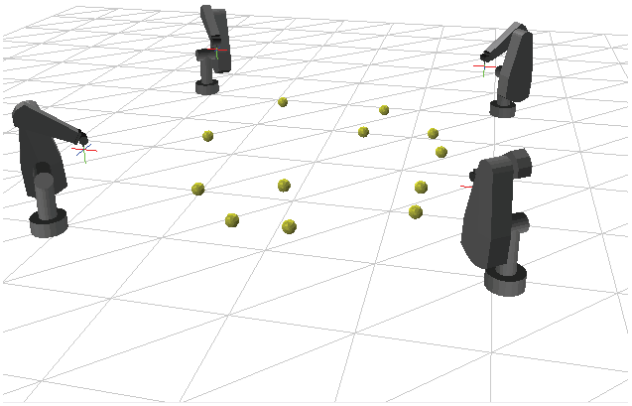


Fig. 2. Task environment

Task 1 requires the robots to pick up the objects (tennis balls) and place them in the centrally-located receptacle (basket). We consider a robot team of 2, 3, and 4 mobile manipulators. The team's performance is measured first with pre-decided division of pick-up targets: each robot is to pick up the objects closest to its initial location, and all robots are assigned with equal numbers of pick-up objects. Next the team's performance is measured with pick-up objects dynamically and spontaneously decided. Table I summarizes the effects of how pick-up is decided, for teams of 2, 3 and

TABLE I

TASK 1 - PRE-DECIDED VS. DYNAMIC (AVERAGED OVER 20 EXECUTIONS)

# robots	*pick-up decision*	*time (m:s)*	*workload distribution*
2	pre-decided	2:36.97	6, 6
	dynamic	2:08.02	5, 7
3	pre-decided	2:14.53	4, 4, 4
	dynamic	1:59.36	4, 4, 4
4	pre-decided	1:59.88	3, 3, 3, 3
	dynamic	1:51.42	4, 3, 3, 2

4 mobile manipulators. The time shown is the total time for the robot team to complete the task. We see that a given team can perform the task quicker when each robot dynamically decides its pick-up target at any time. The optimization criteria for a robot's motion will steer each robot to target an object that is close by and is not obstructed by other robots. We also see that the work load is more or less evenly divided. The performance data confirm that our dynamic and spontaneous task division is more effective than pre-dividing the task.

We also see in table I that increasing the team size reduces the total time to complete the task, but this effect diminishes as the team grows. This is due to the increased crowdedness of the environment by each additional robot, which contributes more moving obstacles to other robots.

Note that the time shown is both the computation time and execution time, as planning and execution are performed simultaneously.

Task 2 requires the robots to pick up the objects and arrange them in a row, as shown in Figure 3. This task is considerably more complex, since each robot must deal with put-down configurations becoming invalid in mid-execution, as well as pick-up configurations. Once a robot has picked up an object and has decided a put-down location (i.e. the next empty spot in the row), that location may become occupied, and therefore invalid, while the robot is in transit. In this case the robot will again choose the next available put-down location, and so on.

Table II summarizes the planner's performance on task 2. We see that the additional requirement of arranging the objects in a row adds considerably to the time required to complete the task. Nevertheless, every team considered still performs the task quicker when each robot dynamically and spontaneously decides pick-up targets than when the task is pre-divided. Again we find that the work load for each robot is more or less even. What our planner achieves very much resembles what a group of human workers will do for the same task. Therefore, our planner confirms that the natural, human way of doing the task is also the more efficient way.

C. Scalability

In a real-world implementation of this distributed method, each mobile manipulator j would be equipped with its own self-contained computer to perform real-time planning of its own motion. The other mobile manipulators will affect

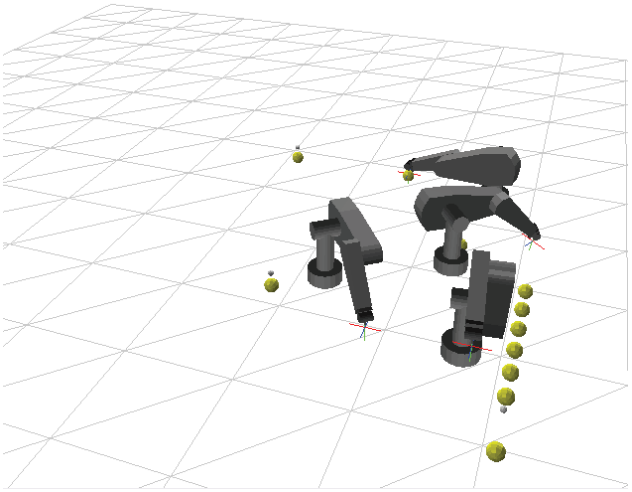


Fig. 3. Four robots about to complete task 2, arranging objects in a row

TABLE II

TASK 2 - PRE-DECIDED VS. DYNAMIC (AVERAGED OVER 20 EXECUTIONS)

# robots	*pick-up decision*	*time (m:s)*	*workload distribution*
2	pre-decided	2:47.16	6, 6
	dynamic	2:38.33	7, 5
3	pre-decided	2:22.78	4, 4, 4
	dynamic	2:11.44	5, 3, 4
4	pre-decided	2:08.03	3, 3, 3, 3
	dynamic	1:49.45	3, 4, 2, 3

the planning efficiency of j only as dynamic obstacles. This resembles well again the case with human workers: in a limited common workspace, there should be a limit on the number of workers before it gets so crowded that the performance of each worker deteriorates. As shown in tables I and II, in our simulated environment, four mobile manipulators can work comfortably with good performance (i.e., reduced working time vs. in the case of fewer robots), where each simulated mobile manipulator is afforded its own processor core. It would be interesting to study the optimal number of workers in a given environment, but that is beyond the scope of this paper.

V. CONCLUSIONS

This paper has introduced a novel approach to real-time, distributed motion planning for a team of mobile manipulators to perform a task together by dynamically and spontaneously dividing the work in a shared environment. Each robot is equipped with its own instance of the same real-time motion planner with the following characteristics:

- The planner achieves real-time adaptiveness by planning path and trajectory together and also by simultaneous planning and execution of motion. This is accomplished by the unique design of the planner and also by exploiting the speed difference between physical motion and computer processing.

- The planner effectively deals with drastic changes in the environment through global planning of diverse trajectories.
- The planner enables dynamic and spontaneous assignment of work efficiently for the robot it serves by allowing each trajectory to adapt the goal pick-up or put-down location based on the availability of valid goals and the optimization of the individual robot motion.
- The planner has the flexibility to incorporate different optimization criteria depending on the need without changing the overall planning algorithm.
- The planner allows the robot to stop either its base or its arm or both in mid-execution of a task when such stopping is more advantageous based on the optimization criteria.

The method is implemented and tested with simulation of mobile manipulator teams in different task environments. The results show the effectiveness and efficiency of the planner and confirm that, by dynamic and spontaneous division of work like human workers do for the same task, the task can be completed more efficiently than predividing the task. Future work includes further testing and improving the algorithm for more complex robots and tasks and incorporating realistic sensing scenarios and constraints. Testing on real robots will also be necessary.

REFERENCES

[1] L. E. Parker, "Alliance: An architecture for fault-tolerant multirobot cooperation," *IEEE Trans. Robotics & Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[2] B. B. Werger and M. J. Mataric, "Broadcast of local eligibility for multi-target observation," *Distributed Autonomous Robotic Systems*, vol. 4, pp. 347–356, 2000.

[3] S. Botelho and R. Alami, "M+: A schema for multi-robot cooperation through negotiated task allocation and achievement," in *Proceedings of IEEE International Conference on Robotics and Automation*, May 1999, pp. 1234–1239.

[4] B. Gerkey and M. J. Mataric, "Sold! auction methods for multi-robot coordination," *IEEE Trans. Robotics & Automation*, vol. 18, no. 5, pp. 758–768, 2002.

[5] A. Farinelli, L. Locchi, D. Nardi, and V. Ziparo, "Task assignment with dynamic perception and constrained tasks in a multi-robot system," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 2005, pp. 1523–1528.

[6] M. Schneider-Fontan and M. J. Mataric, "Territorial multi-robot task division," *IEEE Trans. Robotics & Automation*, vol. 14, no. 5, pp. 815–822, 1998.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

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

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

[19] 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.

[20] T. Yoshikawa, "Manipulability of robotic mechanisms," *International Journal of Robotics Research*, vol. 4, no. 2, April 1985.

[21] Q. Huang, S. Sugano, and I. Kato, "Stability control for a mobile manipulator using a potential method," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 1994, pp. 839–846.

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

[23] F. Schwarzer, M. Saha, and J.-C. Latombe, "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments," *IEEE Trans. Robotics*, vol. 21, no. 3, pp. 338–353, June 2005.