Real-time Planning of Mobile Manipulation in Dynamic Environments of Unknown Changes

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 presents an approach to plan in real-time trajectories of a mobile manipulator in an environment of dynamic obstacles of unknown motions. It exploits loose-coupling of locomotion and manipulation, taking advantage of the redundancy, to best achieve avoidance of dynamic obstacles and various optimization objectives such as minimizing energy and time, and maximizing manipulability. The approach has been implemented and tested in simulation, and the results demonstrate its effectiveness.**

***Index Terms*— mobile manipulator, loosely-coupled locomotion and manipulation, real time, adaptive, dynamic obstacles of unknown motion, partially specified goal, trajectory optimization**

I. Introduction

The combination of mobility and manipulation capability makes a mobile manipulator applicable to a much wider range of tasks than a fixed-base manipulator. For a mobile manipulator, a task goal state is often partially specified as either a configuration of the end-effector, which we call a *place-to-place* task, or a desired path (or trajectory) of the end-effector, which we call a *contour-following* task, and the target location/path of the base is often unspecified. Here a major issue of motion planning is the coordination of the mobile base and the manipulator. This issue, as it involves redundancy resolution, presents both challenges and opportunities.

There exists a rich literature addressing this issue from many aspects. Some researchers treat the manipulator and the mobile base together as a redundant robot in planning its path for place-to-place tasks (e.g, [1], [2], [3]). Some focused on planning a sequence of "commutation configurations" for the mobile base when the robot was to perform a sequence of tasks [4], [5] subject to various constraints and optimization criteria. Others focused on coordinating the control of the mobile base and the manipulator in a contour-following task [6], [7] by trying to position the mobile base to maximize manipulability. Many considered nonholonomic constraints.

While most of the existing work assumes known environments with known obstacles for a mobile manipulator, a few researchers considered local collision avoidance of unknown, moving obstacles on-line. For contour-following tasks, Brock *et. al.* [8] devised an efficient method for the base to adjust its path to avoid a moving obstacle if possible while keeping the end-effector following a contour, such as a straight line. Ögren *et. al.* [9] also introduced a potential-field method for the mobile base to avoid unknown obstacles. Tan and Xi [7] allowed the base to pause in order to let an unexpected obstacle pass while the arm continued its contour-following motion under an event-based control scheme. Mbede *et. al.* [10] used a neuro-fuzzy controller to modify the base motion locally to avoid a moving obstacle stably.

In this paper, we consider real-time motion planning for a mobile manipulator to perform general place-to-place tasks in an environment with many dynamic obstacles of unknown motions, which can obstruct either the base or the arm or both. We present a real-time and global planning approach for simultaneous path and trajectory planning that makes the trajectory of the mobile manipulator in terms of *loosely-coupled* trajectories of the manipulator and the base. The main advantage of loose-coupling for place-to-place tasks is its flexibility in dealing with uncertain dynamic environment: when a dynamic obstacle shows up, sometimes an agile arm movement makes a better avoidance motion than a heavy base movement; sometimes it is more efficient to vary the motion of the base rather than having a large motion of the arm (especially if a large payload is held by the end-effector); sometimes both need to move to make an avoidance; and sometimes it is more energy and even time efficient to just stop the entire system for some period to let the obstacle pass. Our planner allows all the above varieties to happen based on the circumstance and subject to a number of global optimization criteria to minimize energy, time, and maximize manipulability of the whole system motion. It extends our earlier work on fixed-base manipulator planning [11]. In this paper, we consider the mobile base to be holonomic, but the general approach applies to nonholonomic systems as well.

The rest of the paper is organized as follows: Section II provides an overview of our planning approach; section III describes how loosely-coupled trajectories are represented and initialized; section IV introduces trajectory evaluation based on a number of optimization criteria. Section V describes the strategies to alter trajectories to produce better ones. Section VI provides implementation results and discusses performance of the planner. Section VII concludes the paper.

II. Approach

One basic premise of our approach is that the planning process 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 complete trajectories, called a *population*. The feasibility and optimality of each trajectory, called *fitness*, is evaluated through an *evaluation function* coding the optimization criteria. Feasibility refers to collision-free and singularity-free, and feasible trajectories are compared for optimality. The initial population of trajectories is generated randomly, which is then improved to a fitter population through iterations of improvements, called *generations*. In each generation, a trajectory is randomly selected and altered by a randomly selected modification operator among a number of different modification operators, and the resulting trajectory is used to replace the worst trajectory (i.e., the least fit) or a randomly chosen one that is not the fittest in the population to form a new generation. The fittest trajectory is always kept in the population. Therefore, the overall fitness of population improves from generation to generation. Each generation is also called a *planning cycle*.

Once at least one feasible trajectory emerges based on the current knowledge of the environment, the robot can start executing the motion. Since one control cycle for the robot is usually much longer than one planning cycle, as the robot moves, planning continues to improve the population 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 ends at a goal configuration. Such updating is done once in each control cycle.

Changes in a dynamic environment are sensed and fed to the planner in each sensing cycle, which lead to updated fitness values of trajectories in the subsequent planning cycles, and unknown motions of moving obstacles are predicted in fitness evaluation of robot trajectories. Our planner predicts the future trajectory of each moving obstacle from its current sensed state (i.e., configuration, velocity, and acceleration) and previously sensed trajectory and checks the 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 population of ever improving trajectories enables the robot 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 infeasible, the robot does not need to stop and replan from scratch; rather the planner often merely needs to switch the robot to a feasible and better trajectory in the population 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.

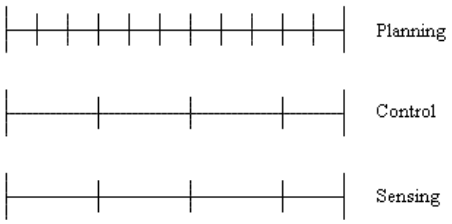

Fig. 1. Relationship among planning, control, and sensing cycles

It should be emphasized that during the simultaneous planning 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 (section IV) so that the change is smooth and stable, and the actual trajectory executed by the robot is the best possible result.

Such planning/control/sensing cycles continue to interact and to move the robot towards a goal configuration in the best possible way in real-time. Figure. 1 illustrates a possible relationship among planning, control, and sensing cycles.

Our approach also supports the partial specification of a goal: only the end-effector position and orientation with respect to the world coordinate system are needed. Different trajectories may hold different goal base configurations and arm configurations (that achieve the same end-effector goal) so that redundancy is exploited to achieve flexibility amid environments with dynamic changes.

III. Trajectory Representation and Initialization

We represent a trajectory of a mobile manipulator uniquely as a pair of manipulator and base sub-trajectories with the following characteristics:

- For the manipulator subsystem, a path of knot configurations is specified in the joint space, based on which a cubic-splined trajectory is used.
- For the base subsystem, a path of knot configurations is specified in its configuration space, based on which a linear-with-parabolic-blends trajectory is used.
- Either trajectory as a function of time may consist of variable segments of constant values, i.e., intervals of no movement, and these time intervals may or may not overlap between the two trajectories. This is the key to achieve loosely-coupled behavior: either subsystem can move while the other does not, or they can move at the same time.
- Two subsystem trajectories are aligned in time to form a trajectory that uniquely determines the motion of the whole system.

Figure 2 illustrates this notion with one joint trajectory and one dimension of the base trajectory.

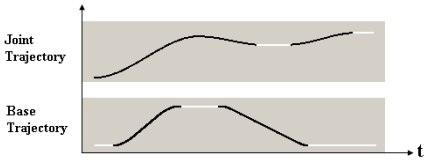

Fig. 2. Loose-coupling of locomotion and manipulation

The number of knot configurations or *knot points* in each sub-trajectory can be arbitrary. Between two knot points is a *trajectory segment*.

To facilitate real-time simultaneous planning and execution, a trajectory always starts from the *current* configuration of the robot. This allows for the seamless transition from one trajectory to another in mid-execution.

An initial population of trajectories can be randomly generated. The robot's initial configuration (including both base and arm) is fully specified and is used as the starting point of each trajectory. However, since only the goal configuration of the end effector is specified, it is up to the planner to find a unique ending configuration for the whole system that is satisfactory. Each trajectory contains a random number of randomly generated intermediate knot points.

IV. 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 our mobile manipulator motion planning problem, we currently use two hard optimization constraints to define feasibility of a trajectory: collision-free and singularity-free. A trajectory is feasible if all trajectory segments are feasible. Otherwise, it is infeasible.

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 [12] of the whole trajectory.

For each infeasible trajectory, we define a fitness value as the sum of a large penalty constant and its fitness function value if it were feasible[1]. Such a fitness function allows us to compare infeasible trajectories and promote improvements of fitter ones among them to be hopefully feasible trajectories. Note that in an initial population, most of the trajectories could be infeasible in an environment of many obstacles.

By defining fitness for not only feasible trajectories but also infeasible ones and by including both types of trajectories in the process for trajectory improvement rather than discarding infeasible ones, our algorithm does not overlook any useful information represented in infeasible trajectories and thus maximizes the efficiency and effectiveness of generating near-optimal feasible trajectories.

It should be noted that in addition to the above criteria, other criteria could be used and aggregated into the evaluation function, requiring changes only in the evaluation procedure, and not to the overall algorithm. We could choose to optimize feasible chromosomes based on any number of criteria, including, for example, safety and stability measures [13]. For non-holonomic mobile manipulators, the non-holonomic constraints could be added as additional hard constraints for evaluating the feasibility of a trajectory and incorporated in the evaluation function for infeasible trajectories.

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 a single 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.

V. Modification Operations

Recall that in each generation of the planning algorithm, certain modification operation is performed on certain trajectories to generate hopefully fitter offspring. We use the following six modification operations:

- **Insert** - a new, random knot point is inserted between two randomly chosen adjacent knot points of a path.
- **Delete** - a randomly selected knot point is deleted from a path.
- **Change** - a randomly selected knot point is replaced with a new, randomly generated knot point.

[1]For a trajectory with singularities, the inverse of its manipulability is infinity. In such a case we use another large penalty in place of the manipulability cost.

- **Swap** - two randomly selected adjacent knot points from a single path are swapped.
- **Crossover** - the knot point lists of two parent 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 arm movement stops at a randomly chosen knot point. 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. We simply randomly select 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 subsystems 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 changes 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.

VI. Implementation, Results, and Discussion

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

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. Both the robot and the objects in the environment are modeled as polygonal meshes for generality. We use the software package [14] to perform real-time collision detection. To simulate environment dynamics, objects are allowed to move in different ways during the trajectory execution; however, the planning algorithm has no a priori knowledge of these movements. As explained earlier, the planning algorithm adapts to the environment dynamics in real time. We implemented the planning algorithm in C++, and the execution is on a Pentium IV 3.2GHz PC.

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 frequency of the control cycle for the 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. The result of this is a surplus of computing power that is useful for real-time motion planning, as the computer has to wait for the mobile manipulator to move anyway. This is exactly as it would be in the real-world case, in place of our simulator, since the speeds of the actuators are very slow as compared to the speed of a modern computer processor.

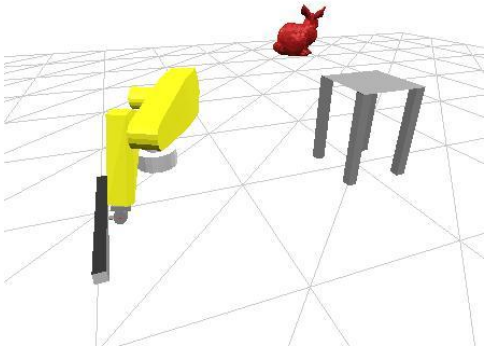

Fig. 3. Task environment 1 (with one dynamic obstacle)

B. Performance Evaluation

We measure the performance of our planner in terms of effectiveness and efficiency. We used many different task environments for testing. Figures 3 through 5 show three different task environments. In task environment 1, the robot's task is to move a long rod from the floor to the table. The table is positioned far enough away that a base movement is required. A moving bunny in mid-air will get into the robot's way, creating a dynamic obstacle. In task environment 2, the task is to grasp the object on the table, and the robot's initial location is far out of reach of the table. There are six moving bunnies: two revolve about the table with different angular velocities, parallel to the floor, and four move in different directions diagonally (i.e., neither vertically nor horizontally but across different altitudes). In task environment 3, the task is the same as that in task environment 2, but the obstacles are different: there are 12 dynamic obstacles of various shapes with changing trajectories; some change their direction and velocity, and some change from linear to angular motion at various times.

In order to measure the optimality of our planner (in terms of energy, time, and manipulability of the trajectories generated), we compare an on-line planned and executed trajectory without knowing obstacle motions with a fully off-line generated trajectory for the same task in the same dynamic environment but with known obstacle motion. We use the same general planner for both on-line and off-line planning. By executing the planner in off-line mode for a very large number of planning cycles (i.e., generations), we produce a trajectory connecting the starting configuration

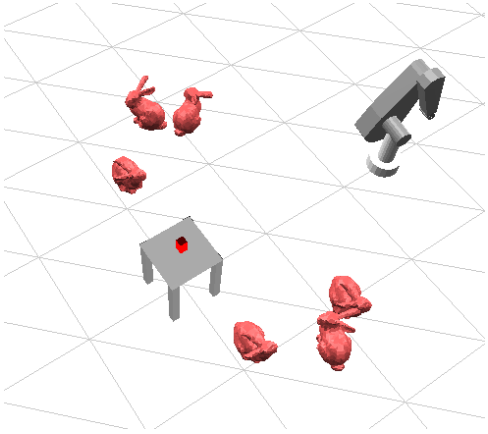

Fig. 4. Task environment 2 (with 6 dynamic obstacles)

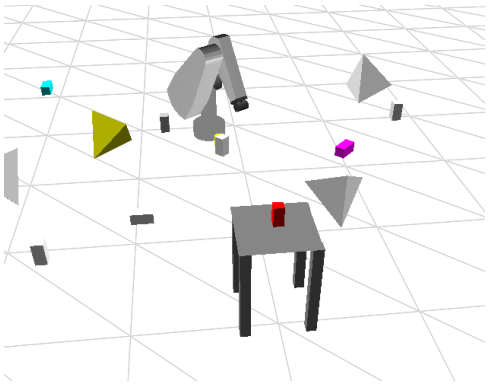

Fig. 5. Task environment 3 (with 12 dynamic obstacles)

and a goal configuration that can be viewed as nearly optimal. In our testing cases, a feasible trajectory can always be found.

Table I compares the results of our real-time planner and those of the off-line planner in each of the the task environments. All the results are the average over 100 executions of each task, and the population size is set to be 20. Column 1 lists the environments as shown in Figures 3 through 5. Column 2 presents the overall cost (i.e., fitness value) of the actual trajectory executed by our real-time planner. Column 3 shows the overall cost of the near-optimal trajectory as the result of off-line planning. Column 4 shows the percent increase of the real-time trajectory cost over the near-optimal trajectory cost, which in fact measures the performance of our real-time planner.

The test results show that a trajectory generated by our planner in real-time carries a cost that is only about 30% higher than that of an off-line planned, near-optimal trajectory. The energy cost E and time cost T are each about 30% higher[2]. Of course, what we gain from this overall higher cost is the ability to deal with unknown environment changes in real-time, which is not possible with off-line planning.

[2]For the three example task environments, the time cost for real-time planned trajectories, which is also the time for execution, is in the range of 10 to 20 seconds.

TABLE I

REAL-TIME VS. OPTIMAL TRAJECTORIES

Task/Env	*Real-time cost*	*Optimal cost*	*% increase*
1	21.03	16.11	23.4%
2	18.67	12.58	32.6%
3	23.88	16.86	29.4%

TABLE II

COMPARISON OF TRAJECTORIES WITH AND WITHOUT STOP OPERATOR

Task/ Env	*Energy Cost (J)*			*Time Cost (s)*		
	w/Stop	*w/o Stop*	*% diff*	*w/Stop*	*w/o Stop*	*% diff*
1	5.56	8.44	-34%	9.22	9.61	-4%
2	4.70	5.65	-17%	10.24	9.91	+3%
3	2.31	3.13	-26%	14.98	16.24	-8%

In order to evaluate the effectiveness of loose-coupling, we compare the results of our real-time planner with the **Stop** operator fully-functional against the results of the planner with the **Stop** operator not used. Table II shows the energy cost E and the time cost T with and without the **Stop** operator, and the percent increase of each. Again, the results are the average over 100 executions the planner, and the population size is set to be 20.

Table II clearly shows the advantages of having the **Stop** operator: the energy costs are down comparing to without the **Stop** operator, and in two cases, the time cost is down as well. The reduction in the energy cost is especially significant. This is because **Stop** permits the mobile manipulator to stop either the arm or the base (or both) to avoid moving obstacles whenever preferred in a non-deterministic fashion (see figure 2) rather than having the mobile manipulator "dancing around" in order to make the same kind of avoidance. The Stop operator saves energy not at the expense of wasting time but, in two of the three environments, also saves time as well, as demonstrated by the test results.

We have tested our planner also in other task environments in addition to the three task environments shown here, and in all cases, the real-time planner was able to avoid all dynamic obstacles with unknown motions and accomplish the task well.

VII. CONCLUSIONS

This paper introduces a novel approach to real-time mobile manipulator motion planning amid dynamic obstacles of unknown motions. The approach has the following characteristics:

- It 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.
- It effectively deals with drastic changes in the environment through global planning of diverse trajectories.

- It has the flexibility to incorporate different optimization criteria depending on the need without changing the overall planning algorithm.
- It produces loosely-coupled locomotion and manipulation trajectories for a mobile manipulator to optimize its motion in performing place-to-place tasks while avoiding unknown obstacle motions.

The method is tested with a simulation of a PUMA 560 mounted to a mobile base in different task environments, with promising results. Future work includes further testing and improving the algorithm for more complex robots and tasks and incorporating realistic sensing scenarios and constraints. Testing on a real robot is also necessary.

REFERENCES

[1] W. Carriker, P. Khosla, and B. Krogh, "Path planning for mobile manipulators for multiple task execution," *IEEE Trans. Robot. Automat.*, vol. 7, no. 3, pp. 403–408, June 1991.

[2] H. Seraji, "A unified approach to motion control of mobile manipulators," *International Journal of Robotics Research*, vol. 17, no. 2, pp. 107–118, 1998.

[3] H. Tanner and K. Kyriakopoulos, "Nonholonomic motion planning for mobile manipulators," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, April 2000, pp. 1233–1238.

[4] F. Pin, J. Culioli, and D. Reister, "Using minimax approaches to plan optimal task commutation configurations for combined mobile platform-manipulator systems," *IEEE Trans. Robot. Automat.*, vol. 10, no. 1, pp. 44–54, February 1994.

[5] D. H. Shin, B.S.Hamner, S. Singh, and M. Hwangbo, "Motion planning for a mobile manipulator with imprecise locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, October 2003, pp. 847–853.

[6] Y. Yamamoto and X. Yun, "Coordinating locomotion and manipulation of a mobile manipulator," *IEEE Trans. Automat. Contr.*, vol. 39, no. 6, pp. 1326–1332, June 1994.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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