2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Hierarchical Planning Architectures for Mobile Manipulation Tasks in Indoor Environments

Ross A. Knepper Siddhartha S. Srinivasa Matthew T. Mason

Abstract— **This paper describes a hierarchical planner deployed on a mobile manipulation system. The main idea is a two-level hierarchy combining a global planner which provides rough guidance to a local planner. We place a premium on fast response, so the global planner achieves speed by using a very rough approximation of the robot kinematics, and the local planner begins execution of the next action even without considering subsequent actions in detail, instead relying on the guidance of the global planner. The system exhibits few planning delays, and yet is surprisingly effective at planning collision free motions. The system is deployed on HERB [20], combining a Segway mobile platform, a WAM arm, and a Barrett hand. The navigation and manipulation components have been tested on the real robot, and the task of simultaneously approaching and grasping a bottle on a countertop was demonstrated in simulation.**

I. INTRODUCTION

We propose to bring indoors a planning architecture that has previously been applied to outdoor mobile robots. Hierarchical planners employ two or more sub-planners that lie on the spectrum from short-range and high-fidelity (local planner) to long-range and low fidelity (global planner). By combining diverse planners, we may take advantage of sensor information at short range to plan detailed trajectories for obstacle avoidance without spending undue computation planning in far away parts of the environment, about which little may be known.

Hierarchical planners afford a number of advantages. The local planner replans frequently, meaning that it is reactive to an unpredictably changing environment. After each replan step completes, the local planner issues a command to the robot for execution during the subsequent planning cycle. Therefore, one observes a minimal amount of lag before the robot starts moving. Finally, the division of labor allows the planner to find paths over great distances and also generate detailed paths which avoid obstacles.

Few planners offer this combination of features. The RRT [15] planner is often highly effective in practice (especially in clutter), however it cannot easily react to a changing environment. By contrast, potential field planners [11] are highly reactive, but they can easily become stuck in cul-de-sacs since they lack global guidance or lookahead capability.

A mature robotic platform, HERB (Fig. 1), hosts our planner implementations. HERB comprises a WAM arm mounted on a differential drive Segway RMP base. HERB localizes itself in real time during navigation with less than 5 cm of error. During manipulation tasks, on-board perception systems localize obstacles and target objects relative to the robot within 1 cm, thus ensuring collision-free trajectories.

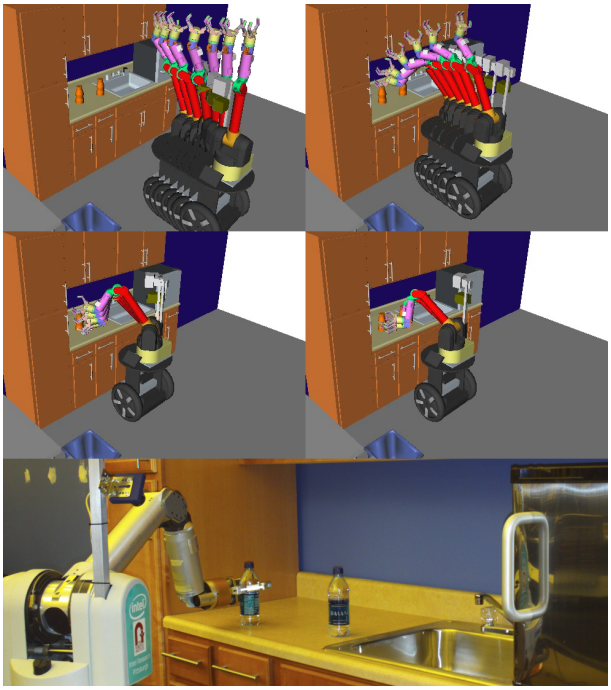

Fig. 1. HERB is a mobile manipulation platform comprising a WAM arm and Segway base. We demonstrate a solution, based on the hierarchical planning framework, for simultaneously driving toward and reaching for objects. Here we depict such an example task, both in simulation, and on the real robot.

We have implemented a pair of hierarchical planners on HERB. A 2D planner handles navigation for the Segway base, while a 3D arm planner performs manipulation tasks. These two planners, in turn, form a combined mobile manipulation planner, in which the 2D planner generates a trajectory for the Segway, which acts as global guidance. Due to the limited reach of the arm, the 3D planner considers shorter distances and acts as a local planner. It receives a predicted trajectory from the Segway planner and generates a path which reacts to its predicted motion. We demonstrate this combined functionality in simulation.

We believe that mobile manipulation is an ideal application for hierarchical planning because of the variation in level of detail. Before manipulating a faraway object, we must initially focus on driving the mobile base. As the target object

R. A. Knepper and M. T. Mason are with The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA - 15213, USA. rak@ri.cmu.edu, mason@cs.cmu.edu

S. S. Srinivasa is with Intel Labs Pittsburgh, 4720 Forbes Ave., Suite 410, Pittsburgh, PA - 15213, USA. siddhartha.srinivasa@intel.com

gets closer, the configuration of the manipulator arm becomes increasingly important, and more time should therefore be devoted to arm planning in the final stage of the operation. With our hierarchical planning architecture, we demonstrate mobile manipulation planning that is spontaneous, effective, and reactive to the environment during execution.

A. Motivation

The hierarchical planning architecture we employ here is designed not for eventual success in the worst-case but for rapid success in the majority of cases. The plans generated by this planner begin running on the robot with minimal planning delay and succeed in reaching the goal under normal circumstances.

In conducting this work, we sought to build a planner possessing a degree of greediness, which can generate goal-directed behavior that is both fast and subjectively "good enough". By simultaneously employing a degree of look-ahead via trajectories in the local planner, this planner may avoid getting stuck in any but the deepest cul-de-sacs in the potential field.

B. Prior Work

The history of hierarchical planners on outdoor fielded navigation systems begins with ALV [5], in which a mapping layer planner provides waypoint guidance for cross-country navigation. The waypoints guide a low-level planner operating in several modes and incorporates several sampled trajectories to select safe vehicle commands.

The design of this early system inspired another outdoor planning architecture in PerceptOR [10], which employs a low-fidelity global planner based on Field D* [8] to give global guidance to a local planner. This local planner evaluates candidate trajectories, generated by sending sampled constant-input controls through a vehicle model. These trajectories terminate at various points in the workspace. All collision-free paths are permissible because D* provides a navigation function, thus giving the route from any point to the goal. The local replan rate is roughly 10 Hz.

Two of PerceptOR's successors, Crusher and Boss [9], employ hierarchical planners in different environments. Crusher concentrates on off-road driving, while Boss, the winner of the 2007 DARPA Urban Grand Challenge, navigates roads. A variety of other Urban Challenge teams also explored the use of hierarchical planners to varying degrees (see [13] for a survey of Urban Challenge local planning solutions).

The Argo platform [1] implements a separate heritage of off-road hierarchical planners. While PerceptOR and its kin generate a global navigation function which can be queried from any point, Argo's global planner offers a single path for the local planner to track. Thus, only local paths which terminate on the global path can be considered. In the present work, the separate navigation and arm planners employ the PerceptOR paradigm, while the combined mobile manipulation planner operates similarly to Argo.

A twist on the hierarchical planning concept, presented in [17] for planetary rovers, combines search spaces of varying fidelity and scalability into a unified graph. This planner performs comparably to many hierarchical planners with a single D* query at the expense of additional bookkeeping to manage the world-fixed search graph as the rover traverses its environment.

Another body of work relates to hierarchical planning for indoor navigation. Candido [4] discusses the application of hierarchical planning to humanoids. Since the high degrees of freedom make exhaustive C-Space planning impractical, this work instead employs a look-up table of feasible motions, called motion primitives, which generate useful behaviors when applied to a terrain. Candido builds a graph of motion primitives, which explore the terrain.

Yang [22] implements a hierarchical planner loosely resembling our architecture. The global planner generates a probabilistic road map with movable nodes to handle dynamic environments. The objective is to maintain an approximation of the changing topology of the environment within this graph. During execution, a potential field local planner follows these graph edges.

The indoor hierarchical planning architecture most closely resembling our own is reported by Wang [21]. In this work, a global planner computes the shortest path in a grid from start to goal. A local planner then tries to follow this path by defining a sub-goal where a circle centered at the current robot position intersects the target path. Potential fields drive the robot toward its sub-goal, thus performing receding horizon control until the robot reaches the final goal.

Handey [16] is a fully integrated system capable of recognizing and manipulating polyhedral parts in clutter. Handey's motion planner is composed of a two-layer hierarchy of a coarse-grained kinematic arm planner and a fine-grained hand controller for executing planned compliant grasping. The intuition for the hierarchy is that while the arm had to merely avoid obstacles, the hand had to reach into tight spaces, make deliberate contact with obstacles, and perform fine manipulation.

Finally, [12] performs some analysis of hierarchical planning for navigation in a theoretical and simulation study.

II. HIERARCHICAL PLANNING

The planners described in this work come from a heritage of planners deployed in the context of outdoor rough-terrain navigation. Vehicle safety requires a high-fidelity vehicle model capable of predicting the vehicle's response to candidate actions over the upcoming ground shape. Model predictive planning and control utilize such a vehicle model to select and steer the best trajectory. Each is typically executed in a loop. In our system, the local planner replans at 5 Hz, while the controller loops more rapidly at 37.5 Hz.

Model predictive planners are somewhat costly to execute since they are limited by the speed of the accompanying vehicle model. A rapid replan rate is important to ensure the most up-to-date sensor data for the upcoming ground patch, and replanning also provides the robot an ability to react to dynamic obstacles in real time. Together, a rapid replan rate and a costly model constrain the amount of available

computation during each replan cycle, which has the effect of bounding the horizon of local planning.

To compensate for this horizon, we typically employ one or more higher level planners that make different tradeoffs. In general, an entire spectrum of planners may be employed in a hierarchy, where the largest scale may involve no more than a heuristic distance estimate. Each successively lower level plans to a shorter horizon while more accurately representing actual vehicle motions.

In this work, we restricted the hierarchy to two levels. At the global level, we run D* Lite [14]. We discretize the workspace into a rectangular grid and run the D* search algorithm to rapidly find the cost of a path from any point in the workspace to the nearest goal position. In searching a grid, we trade off guaranteed trajectory feasibility for computational speed. D* offers additional advantages over other graph search algorithms such as reuse of previously generated results and the ability to alter cell costs due to dynamic obstacles without the need to plan from scratch.

At the local level, the planner generates a set of paths which are feasible to execute on the target system. Given paths τ each parametrized as $\tau : [0, 1] \rightarrow Q$, these paths are scored with a heuristic cost function of the form

$$c_\tau(\tau) = c_{gp}(x(\tau(1))) + c_{lp}(\tau) + \alpha^T f(\tau), \qquad (1)$$

where q is a robot configuration, $x(q)$ is the corresponding workspace coordinate, $f(q)$ is a vector of features on the robot configuration, and α is a vector of feature weights, which may be hand-tuned or learned using a technique like logistic regression. The global and local path costs are represented by $c_{gp}(x)$ and $c_{lp}(\tau)$, respectively.

In the remainder of this paper, we discuss two indoor applications for the hierarchical planner. In Section III, we address indoor navigation of a Segway RMP based robot in an office environment. In Section IV, we extend the approach to planning for the Barrett WAM arm. Taken together, these planners represent steps toward a unified hierarchical model predictive mobile manipulation suite.

III. 2D NAVIGATION PLANNER

In designing a navigation system for HERB's Segway base, we utilized ROS [18]. Just as with the architecture described above, we divided the navigation planner into local and global planners with a heuristic function arbitrating between them. The navigation planner is based heavily on the planner described in [12]. The local planner generates a set of polynomial-parametrized curves in action space with a six-second lookahead.

After selecting the best path at each cycle, the planner passes the corresponding command to the robot for open-loop execution. We produced a high-fidelity dynamic model of the Segway based on data collected from HERB driving. We use this model to predict the trajectory resulting from a given control. The predictive model of the Segway is sufficiently accurate as to introduce negligible error over the course of a fraction of a second between commands. Consequently, we do not require a path-following controller.

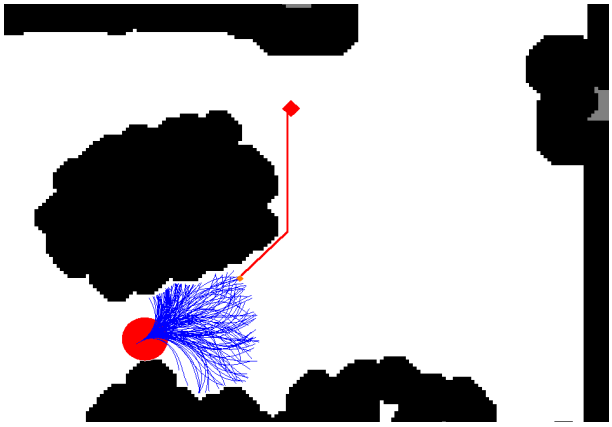

Fig. 2. Navigation planner view showing local candidate curves in blue and the straighter global D* path (in red) connecting the best local path to the goal (red diamond). Obstacles are in black.

A. Local Planner and Controller

The local planner replans at a rate of 5 Hz. During each cycle, it generates a fixed number of randomly-sampled trajectories. Paths are represented by polynomial functions of linear and angular velocity controls. We resample the paths during each cycle in order to customize their selection and tune performance. For long-range navigation, we have found that restricting the set of controls to fixed velocity and limited curvature (i.e. a car-like motion model) produces smoother motion than the more general set of differential drive actions. By contrast, when the vehicle is near the goal, the full range of differential drive motions becomes crucial to success. Rather than abruptly switching motion types, we sample variably from two distributions on controls (car-like and differential drive) according to goal proximity.

We test each predicted path for collision with obstacles and compute a heuristic cost function on each trajectory. We then pass the lowest-cost control to the controller for execution.

Since we are operating indoors on flat surfaces, the vehicle model is sufficiently good that we can drive the Segway open-loop for 1/5$^{\text{th}}$ sec at a time. Therefore, the controller's main job is merely to pass commanded velocity trajectories to the robot. However, it also monitors HERB's SICK ladar, broadcasting at 37.5 Hz, for imminent collisions, which are detected by simulating forward the last command beyond the stopping distance. If the controller's "virtual bumper" predicts a collision, then the commanded linear velocity is overridden with zero speed until the hazard clears. We put the controller infrastructure in place to guarantee highly reactive safe behavior, but the virtual bumper is rarely necessary since the local and global planners will react to obstacle movements at 5 Hz.

B. Global Planner and Heuristic Function

In the navigation planner, D* Lite plans in a map where obstacles are expanded by the radius of the robot. The end point of each local trajectory becomes the starting point for the global path, and the sum of the two path costs (i.e. the line integral of a path with a costmap) gives the total path

cost. D* is suitable for use in a real time environment. In an 8-connected grid, our D* implementation expands cells at a rate of 300,000 cells per second, consuming about 0.1 second amortized over a whole run.

The navigation planner's vector of configuration features from (1) consists of

$$f_{nav}(\tau) = [c_{ang}(\tau(1)) \quad c_{prox}(\tau) \quad c_{sim}(\tau)]^T. \quad (2)$$

Here, $c_{ang}(q)$ returns the cost resulting from the difference in heading between the terminal configuration in the local path and the gradient of the global navigation function. $c_{prox}(\tau)$ denotes the cost reflecting the proximity of the path culminating in q to any obstacle. Finally, $c_{sim}(\tau)$ reflects the similarity between the commanded action ending at q and the currently-executing action: given two paths of equal cost, we prefer the path that minimizes acceleration.

C. Results

During manual tuning, we found that the feature weights are not very sensitive, and so good performance is easy to attain in practice. Following tuning, we tested the navigation component extensively on HERB. The robot reliably planned in real time, successfully navigating around an office environment with large and small spaces for a thirty-minute stress test without getting stuck or striking an obstacle. The robot accomplished this feat while continuously updating its map to reflect movement of people and furniture in the environment. The navigation planner is now in use on two different projects employing HERB for other tasks.

IV. 3D ARM PLANNER

In this section, we move to the problem of applying a hierarchical planner to a manipulator arm. We executed our planned trajectories on a WAM arm from Barrett Technology, Inc. with seven degrees of freedom. We utilized the OpenRAVE [6] planning environment, which provides kinematics, collision checking, and visualization capabilities.

A. Problem Description

As manipulation represents a huge scope for ongoing research, we constrained the task to reaching from a start configuration to any one of a set of goal poses of the end effector. We exploit the affordance offered by the manipulation problem, allowing the hand to approach a target object from any free direction. In fact, we generalize even further by allowing the planner to choose among multiple target objects.

The model we used for goal specification is a simplified form of TSR (Task Space Region [3]) for specifying goal neighborhoods. A TSR describes a range of valid end effector poses. For example, a bottle may be grasped from any side with a small translational tolerance. For this work we kept the implementation simple by instead specifying a small set of goal poses sampled from the TSR. From these goal poses, we compute a corresponding set of inverse kinematic solutions using OpenRAVE's IKfast solver.

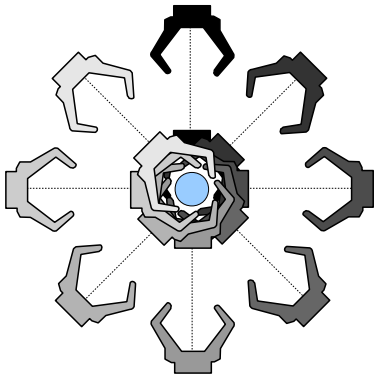

Fig. 3. Sampling multiple target goal poses around an object. For small finger apertures, the goal state (ready to grasp the object) may be hard to reach. Therefore, we pull back by a small amount and generate a penultimate goal, from which the ultimate goal is reached without search.

Depending on the finger aperture, it may be difficult to find any configuration from which the goal is visible in joint space. This form of the narrow passage problem is easily solvable because we know that pulling the hand straight back from the target object is a valid motion (pending necessary collision checks). Therefore, by designating a set of penultimate goals set back a short distance from the true goal, a solution becomes easier to find (Fig. 3).

B. Local Planner

The arm controller relies on a combination of model predictive and PID control. We found this controller to be sufficiently reliable that we could generate purely kinematic plans and count on the controller to execute them accurately.

As with the Segway planner, we allow the arm to begin moving toward the goal before the entire plan has been generated. Planning iterations complete at a rate of about 5 Hz on average, just as on the Segway. Although paths are planned much faster than they can be executed, we cannot replace old commands with new ones as we did in the 2D planner because we currently lack the capabilities both to predict where the arm will be a fraction of a second hence and to interrupt an executing command. Thus, we execute all paths to their endpoints. Fig. 4 shows a visualization of all the searched paths through a series of steps culminating in reaching the goal.

During each replan step, the local planner samples a fixed number of straight-line motions in joint space. These paths comprise two categories. To produce a degree of greedy goal-directedness, we generate several paths which move in a straight line in joint space from the current configuration to the goal configurations. For the sake of exploration, the remaining motions are sampled from a random distribution. We represent all paths as relative motions in joint space, thus they are not particular to a given start configuration.

In the event that one of the goal-directed paths is collision-free, the arm immediately moves to the goal and returns success. However, such goal path collision tests often fail because each target grasp is inherently in close proximity to

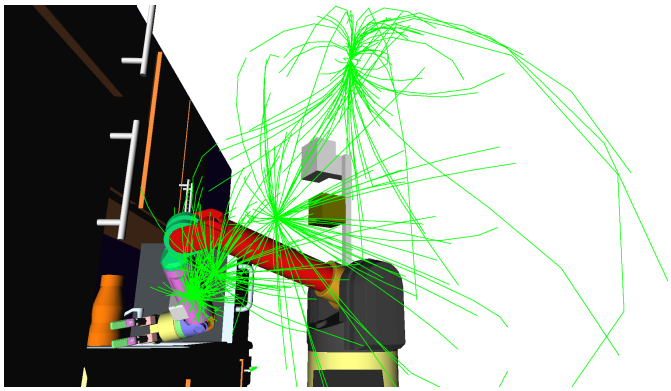

Fig. 4. An example plan to the goal, showing all considered trajectories. The arm began in singularity at the top of the figure, proceeding down and left toward the counter. It is shown near the final configuration, about to grasp the juice bottle.

one of the target objects. In this case, we retain colliding paths by truncating them short of the distance at which the collision occurred (minus 20% of the total path length).

We generate the majority of paths by sampling relative motions from a random distribution. Since there is no guarantee that these paths will be useful, we lazily collision test only those paths which are candidates for traversal. If one of the random paths has the lowest cost, then it will be tested for collision with the environment, self-collision with the robot, and exceeded joint limits. A path failing any of these tests is eliminated from consideration (as with the 2D planner).

The path sampling distribution was the subject of some engineering. We break the sampling process into two stages. In stage one, we sample directions in C-space. Then in stage two, we scale each path according to a distribution over lengths that is a function of the remaining distance to the nearest goal. This separation enables the planner to explore widely at the beginning and then gradually move to fine-tuning its position as the arm approaches a solution. We achieve this behavior without fundamentally altering the path generation algorithm at any point during planning.

We initially tried picking directions by uniformly sampling points from an n-dimensional hypersphere. Such algorithms are somewhat computationally expensive. To improve performance, we next tried sampling from the circumscribing n-hypercube. This fast operation gives a distribution biased toward motions where C-space velocity is better distributed throughout the joints of the arm (corresponding to the corners of the hypercube). In practice, this distribution appears to better explore the C-space in fewer steps, particularly in cases where the arm starts from a singularity, as in Fig. 4

After generating a unit-length vector, we then scale it based on a Gaussian random variable. Both the mean and standard deviation of this distribution are equal to half the C-space distance to the goal, d_i. We scale each candidate path to a joint space length, s_i as

$$s_i \leftarrow trunc\left(N\left(\frac{d_i}{2}, \left(\frac{d_i}{2}\right)^2\right), [0, \infty)\right). \quad (3)$$

We apply a heuristic scoring function to all paths and then pass the lowest-cost path that survives the collision test to the arm for execution.

C. Global Planner and Heuristic Function

As with the 2D planner, we employ D* Lite to generate a navigation function in 3D workspace position for the end effector, planning in a 26-connected grid. Note that we are treating the hand as if it is floating in space; the resulting D* paths do not reflect potential collisions between the arm and environment. As a consequence of this trade-off, performance is quite good. In 3D, D* performs over 100,000 expansions per second. Due to the compact workspace, D* expends a mere 1/15$^{\text{th}}$ sec on a typical reaching problem.

The arm planner's vector of configuration features from (1) consists of

$$f_{arm}(\tau) = \left[c_{df}(\tau) \quad c_{cs}(\tau(1)) \quad c_{jl}(\tau(1))\right]^T. \quad (4)$$

The cost $c_{df}(\tau)$ comes from a squared distance field akin to the signed distance field of [19]. This distance field maps each workspace cell to the squared distance to the nearest object in the world. The computation is efficient (0.06 sec for 1 million voxels on a standard desktop processor). To find a distance field cost, we discretize the arm lengthwise and step through to find the point of nearest contact for the arm. Conveniently, the arm approximates a constant-radius cylinder, so we may readily compute the workspace distance, $d_{ws}(\tau)$. The cost $c_{df}(\tau)$ is computed as

$$c_{df}(\tau) = \frac{1}{d_{ws}(\tau)^2}. \quad (5)$$

We measure the straight-line C-space distance to the closest goal configuration, $c_{cs}(q)$. This term may appear redundant since $c_{gp}(x)$ also measures distance to the goal, but the two are complementary. $c_{cs}(q)$ is aware of kinematics but ignorant of obstacles, while $c_{gp}(x)$ incorporates obstacles but not kinematics. Since the optimal path is both kinematically feasible and obstacle-free, we use a weighted sum of the two terms. Inspiration for this approach comes from [7].

We compute $c_{jl}(q)$ based on the distance of each joint to the nearest joint limit. It is desirable to keep the arm away from joint limits both for manipulability and equipment health. Given the distance d_j of joint j to the nearest limit, the joint limit cost is computed as

$$c_{jl}(q) = \sum_j \frac{1}{d_j^2}. \quad (6)$$

D. Results

We tested the arm planner in a kitchen environment. The task, pictured in Fig. 1, is to approach and grasp any one of several bottles on the table. We compared the standalone hierarchical arm planner, running on the real robot, against a state of the art bidirectional RRT-based planner, CBiRRT [2]. CBiRRT is capable of accepting a list of TSRs describing object grasps, hence it is solving a similar problem to our own planner. In performing a comparison, we examined

TABLE I

COMPARISON WITH BIDIRECTIONAL RRT AVERAGED OVER 5 TRIALS.

Planner	Average onset of motion (sec)	Average time to goal (sec)	Average collision checks
Hierarchical Planner	0.2	12.7	382
CBiRRT	2.4	14.7	1755

three planning metrics. First, we measured the delay between planning onset and the start of motion. Second, we measured the total task time from start of planning to reaching the goal. Third, we looked at the total number of collision checks each planner performed. Typically, collision checks strongly influence planning time because collision checking is quite expensive. Table I shows the results of this comparison.

V. DISCUSSION AND FUTURE WORK

This paper describes preliminary work on hierarchical planning for mobile manipulation. We have reported results on a real robot for the decoupled navigation and arm planning problems. We also demonstrated the combined mobile manipulation planner in simulation (Fig. 1). A thorough examination of the coupled mobile manipulation planner running on HERB remains a subject of future work. However, several other areas of future work present opportunities.

Bidirectional planners have been popular for a decade, particularly in the RRT community. In interleaved planning and execution, bidirectionality has a different meaning since the robot can only execute in the forward direction. However, we are already getting some benefit from bidirectional planning, as we are sampling pullback goals (Section IV-A), from which we know how to reach the ultimate goal. We believe this idea can be generalized to discover regions of space from which a goal state is most reachable.

At present, the hand approaches a goal position in any arbitrary orientation, but we could sample in fewer dimensions and constrain the palm or wrist to point towards a goal or other nearby object. Then, as the hand moves closer to its goal, we can use distally-mounted cameras to localize the target, providing improved position estimates while planning.

Finally, we would like to explore the planner's failure modes in hopes of designing a global planner which succeeds more often. Occasional failure is a fair price to pay for a planner that works quickly and generates reasonable paths under most circumstances. In addition to these benefits, the hierarchical planner offers reactivity to dynamic obstacles and nimble maneuvering in cluttered environments.

VI. ACKNOWLEDGMENTS

This work is sponsored by the Defense Advanced Research Projects Agency. This work does not necessarily reflect the position or the policy of the Government. No official endorsement should be inferred.

This material is based upon work partially supported by the National Science Foundation under Grant No. EEC-0540865.

REFERENCES

[1] T. Allen, J. Underwood, and S. Scheding. A path planning system for autonomous ground vehicles operating in unstructured dynamic environments. In *Proc. Australasian Conf. on Robotics and Automation*, 2007.

[2] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner. Manipulation planning on constraint manifolds. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, May 2009.

[3] D. Berenson, S. Srinivasa, D. Ferguson, A. Collet Romea, and J. Kuffner. Manipulation planning with workspace goal regions. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, May 2009.

[4] S. Candido, Y. Kim, and S. Hutchinson. An improved hierarchical motion planner for humanoid robots. In *Proc. IEEE-RAS Internaional Conf. on Humanoid Robots*, Daejeon, Korea, December 2008.

[5] M. Daily, J. Harris, D. Keirsey, D. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous cross-country navigation with the ALV. In *Proc. IEEE Intl. Conf. Robotics and Automation*, Philadelphia, PA, 1988.

[6] R. Diankov and J. Kuffner. OpenRAVE: A planning architecture for autonomous robotics. Technical report, Robotics Institute, Carnegie Mellon University, July 2008. tech. report CMU-RI-TR-08-34.

[7] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous driving in unknown environments. In *Proc. Intl. Symp. on Experimental Robotics*, Athens, Greece, July 2008.

[8] D. Ferguson and A. Stentz. Field D*: An interpolation-based path planner and replanner. In *Proc. Intl. Symp. on Robotics Research*, October 2005.

[9] T. Howard, C. Green, D. Ferguson, and A. Kelly. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *J. Field Robotics*, 25(1):325–345, June 2008.

[10] A. Kelly, A. Stentz, O. Amidi, M. W. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. M. Vallidis, and R. Warner. Toward reliable off road autonomous vehicles operating in challenging environments. *Intl. J. Robotics Research*, 25(1):449–483, May 2006.

[11] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Intl. J. Robotics Research*, 5(1):90–98, 1986.

[12] R.A. Knepper and M.T. Mason. Empirical sampling of path sets for local area motion planning. In *Proc. Intl. Symp. of Experimental Robotics*, Athens, Greece, July 2008.

[13] R.A. Knepper and M.T. Mason. Path diversity is only part of the problem. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, Kobe, Japan, May 2009.

[14] S. Koenig and M. Likhachev. D*Lite. In *AAAI/IAAI*, 2002.

[15] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *Intl. J. Robotics Research*, 20(5):378, 2001.

[16] T. Lozano-Perez, J. Jones, E. Mazer, P. O'Donnell, W. Grimson, P. Tournassoud, and A. Lanusse. Handey: A robot system that recognizes, plans, and manipulates. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 4, 1987.

[17] M. Pivtoraiko and A. Kelly. Differentially constrained motion replanning using state lattices with graduated fidelity. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, September 2008.

[18] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[19] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, May 2009.

[20] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe. Herb: A home exploring robotic butler. *Autonomous Robots*, 28(1):5–20, 2010.

[21] L.C. Wang, L.S. Yong, and M.H. Ang. Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment. In *Proc. IEEE Intl. Symp. on Intelligent Control*, Vancouver, Canada, October 2002.

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