A Hierarchical and Adaptive Mobile Manipulator Planner

Vinay Pilania and Kamal Gupta

***Abstract*— We present a Hierarchical and Adaptive Mobile Manipulator Planner (HAMP) that plans for both the base and the arm in a judicious manner - allowing the manipulator to change its configuration autonomously when needed if the current arm configuration is in collision with the environment as the mobile manipulator moves along the planned path. This is in contrast to current implemented approaches that are conservative and fold the arm into a fixed home configuration. Our planner first constructs a base roadmap and then for each node in the roadmap it checks for collision status of current manipulator configuration along the edges formed with adjacent nodes, if the current manipulator configuration is in collision, the manipulator C-space is searched for a new reachable configuration such that it is collision-free as the mobile manipulator moves along the edge. We show that HAMP is probabilistically complete. We compared HAMP with full 9D PRM and observed that HAMP outperforms the full 9D PRM in each of the performance criteria, i.e., computational time, percentage of successful attempts, base path length, and most importantly, undesired motions of the arm.**

I. INTRODUCTION

Autonomous Mobile Manipulation is an important problem for humanoids, particularly in service robotics applications. It includes several intertwined sub-problems including finding suitable grasps and grasping the object [1], [2], finding the best mobile base location and manipulator configuration corresponding to the end effector pose [3], close-range scene segmentation for table-top manipulation [4] with static mobile base, and finding the mobile manipulator path from start to goal (base poses and manipulator configurations) [5]. In this paper we focus on the last sub-problem, i.e., determine a collision-free path for the mobile manipulator from a given start configuration to a desired goal configuration. Most work [5], [6], [7] usually takes a very conservative approach, which is, to fold the arm to some safe "home" configuration and then plan for a 2D projected footprint of the mobile manipulator in a projected 2D representation of the world from start base pose to goal base pose.

Clearly, this approach has two main limitations: (i) the projection of the mobile manipulator with extended arm may have a large footprint, and may be in collision with 2D projected map, while the mobile manipulator is collision-free in 3D map, and more fundamentally (ii) it may not always possible to change the arm to a predefined home configuration at base's start pose because of physical constraints or there may be task constraints that prevent the arm being folded, e.g., if the robot is carrying a glass of liquid which needs to be kept vertical to avoid spillage. Another example is where the mobile manipulator is carrying a long payload, say a pole and it needs to continuously move the arm (and thereby the pole to avoid the pole colliding with walls and other objects in the environment) to navigate through the doors and hallways. In such scenarios, mobile manipulator with arm in start configuration can not reach the goal unless it changes the arm configuration several times along the path. One such example is shown in Figure 1.

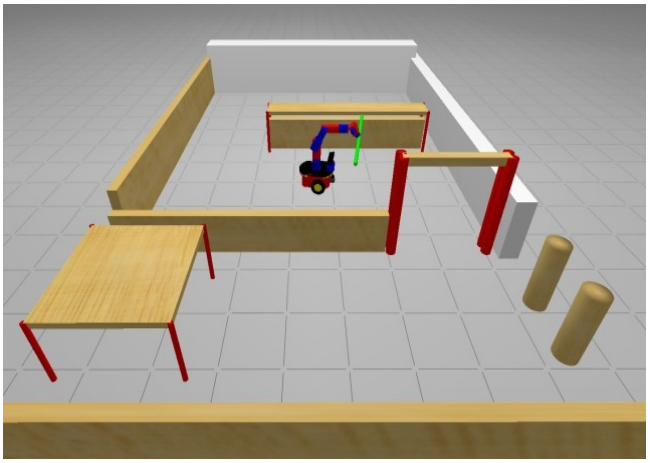

Fig. 1. This example (simulation environment for scenarios A and B in Section V) shows the mobile manipulator carrying a long payload (120 cm long stick) and it needs to pass through a doorway to reach the goal on the other side.

One possible solution to this motion planning problem is to use sampling based planners [8], [9] in full configuration (C-space) of the mobile manipulator. However, besides being somewhat computationally expensive, the computed path for the mobile manipulator may result in undesired and excessive motions for the manipulator. This is primarily because of the randomness associated with sampling based planners and persists even after applying a post processing smoothing filter. In most scenarios, there is no need to move manipulator except at certain base poses - the undesired arm motion (post smoothing) refers to this extraneous manipulator motion while the base is moving. We would like to avoid such undesired manipulator motions.

We propose a Hierarchical and Adaptive Mobile Manipulator Planner (HAMP) to solve the problem and a schematic illustrating the planned mobile manipulator path is given in Figure 2. The HAMP algorithm is a two stage process: in the first stage it constructs a base roadmap (using PRM in the base configuration space) where it connects the start and goal base poses (with manipulator remaining in a fixed

Vinay Pilania and Kamal Gupta are with Robotic Algorithms & Motion Planning (RAMP) Lab, School of Engineering Science, Simon Fraser University, Burnaby, BC V5A1S6, Canada. Email: vpilania@sfu.ca, kamal@sfu.ca

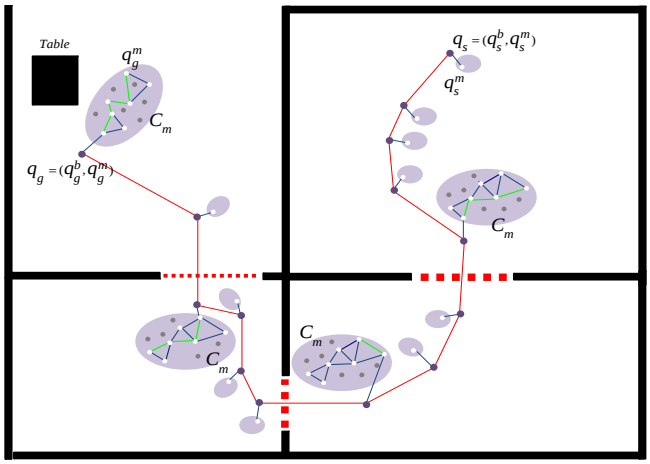

Fig. 2. A schematic illustrating the planned mobile manipulator path Π^{bm} given by HAMP algorithm. Please see text for explanation.

home[1] configuration). In the second stage, the algorithm reconfigures or "adapts" the manipulator configuration to a new configuration along the edges in the base roadmap constructed earlier by checking for the manipulator collisions along them from start to goal. This two stage process iterates until a collision-free mobile manipulator path is found or the time limit is over. The second stage works as follows: for each node in the base roadmap, the current manipulator configuration is checked for collisions along the edges corresponding to the adjacent nodes. If it is in collision along an edge in the base roadmap, then the manipulator is reconfigured (while base is stationary at the base node) by moving it to a new configuration such that the new configuration is collision-free if the mobile manipulator with manipulator in new configuration were to move along the edge in the base roadmap. This reconfiguration step is carried out via motion planning for the manipulator in the manipulator's C-space constructed at the given base node. If no such manipulator configuration is found, then a new edge will be searched for in the base roadmap, and the process repeats. Figure 2 schematically illustrates HAMP algorithm. In the figure, blue dots correspond to base pose nodes, the red segments are the base edges, and light purple ellipses (small and big) corresponding to each blue dot is the manipulator C-space. Small purple ellipses with one white dot indicate that the manipulator configuration, corresponding to the white dot, is free along the base edge (to the next base node) and no manipulator planning was required. Three red color dash lines denote the physical gates (overhead view). The big ellipses show where manipulator planning was done, with the manipulator roadmap shown with its nodes and edges inside each ellipse. For the first three ellipses, the manipulator configuration at each base node just before the gate was in collision along the edge (as the mobile manipulator moves through the gates) and hence the roadmap was built and searched for a path and the sequence of light green edges shows the path. The manipulator moves along this path to the end configuration, which is, by construction (as explained in the detailed algorithm in Section IV), collision-free as the mobile manipulator moves across the gate to the next base node. The fourth big ellipse (at base goal pose) shows a reconfiguration step to the goal configuration of the manipulator.

Summarizing, HAMP searches in two sub-spaces (base sub-space and manipulator sub-space) in a novel way and on a "need to" basis, i.e., the search in manipulator space is invoked only for those points in the base-space where it is needed. Hence, HAMP searches a much smaller size of space, as a result, it computes paths in shorter time with higher success rate than a search in the full configuration space of the mobile manipulator, and more importantly, it also avoids unnecessary motion of the arm, as is the case for the full search. Both these key points are validated in our experiments. Furthermore, we have a proof that HAMP is probabilistically complete. Because of space limitation, the proof is not included in the paper and will be included in a subsequent journal version. It can be accessed online at `http://www.sfu.ca/~vpilania/research.html`.

We emphasize that while we chose to use PRM as underlying core sub-planner (for base and for the manipulator) within the HAMP framework, one could easily substitute other planning schemes such as Numerical Potential Field [10] for the base or other sampling based variations such as RRT [8] or Bi-directional RRT [11] as the underlying core sub-planners within the HAMP framework. In fact, our PRM implementation of core sub-planners is an adaptation of the incremental version available in the library OMPL [12]. Our contribution is the novel HAMP search framework.

II. RELATED WORK

Most of the previous work [13] [14] [15] [16] on mobile manipulation mainly deals with the coordination of the mobile base and the manipulator motion for following a given end effector trajectory. In motion planning related work, Hornung *et al.* [17] improved upon [6], [7] using a multi-layered 2D representation of both the robot and the environment. However, since the planning is still carried out only for the base and not the manipulator, their approach will fail in the scenarios where arm configuration needs to be changed while navigating from start to goal. Hornung and Bennewitz [18] proposed an adaptive approach for efficient humanoid robot navigation, which allows for finding solutions for foot-step planning where planning based on a 2D grid fails. Our approach has a similar adaptive flavour, but it is in the context of mobile manipulation and not foot step planning. In the context of mobile manipulators, hierarchical strategies have been used to estimate reachable workspace [19]. An interesting use of adaptive dimensionality has recently been introduced in [20]. Their approach uses deterministic search (A* over discretized C-space) in a low dimensional end-effector C-space interleaved with tracking in the full mobile manipulator C-space. It is shown that the resulting planner outperforms a full dimensional RRT in a class of tasks

[1]Note that there are other options here, e.g., one could simply construct the base roadmap for the base only, however, this could lead to several nodes/edges being invalidated in the subsequent stage.

where the end-effector is carrying a large payload. One could characterize this approach toward the "greedy" end of the spectrum since the search is, in effect, guided by a path for the end-effector. While this approach could be used in a relatively small region near the goal, as shown in the example tasks in the above mentioned paper, a key problem is that due to its deterministic search, it is not applicable to relatively large areas as is the case in our examples. Finally a genetic optimization based planner for a mobile manipulator that plans motions in full configuration space in dynamic environments is presented in [21]. Note that the performance of genetic optimization relies on maintaining a diverse population of trajectories that belong to different homotopic groups which is a significant challenge.

III. PROBLEM FORMULATION

We use $q_i = (q_i^b, q_i^m)$ in C_{bm}, the C-space of the mobile manipulator, to represent i^{th} mobile manipulator configuration, where $q_i^b = [x, y, \theta] \in C_b$, the C-space of the mobile base, is the base configuration (also called base pose) and $q_i^m = [\theta_1, \theta_2, ..., \theta_d] \in C_m$, the C-space of the d degree of freedom manipulator, is the manipulator configuration. $C_{b_{free}}$ is the set of all collision-free base poses and $C_{b_{obs}}$ is the set of poses resulting in collision with obstacles. For a given base pose, q_i^b, $C_{m_{free}}$ denotes the set of free manipulator configurations (for simplicity we omit the reference to the corresponding base node q_i^b in the notation) and $C_{m_{obs}}$ denotes the set of manipulator configurations that are in collision with obstacles. We use q_H^m to denote the home configuration of the manipulator, a compact and folded configuration of the arm, specified by the user. For simplicity of explanation, the 3D environment is assumed to be known or acquired by previous sensing, but our framework is extended to the simultaneous sensing and planning by incorporating a view planner similar to [22], [23]. Given a 3D map, the start $q_s = (q_s^b, q_s^m)$ and goal $q_g = (q_g^b, q_g^m)$, the objective of our HAMP algorithm is to find a collision-free path.

Because of the hierarchical and adaptive approach, the nature of mobile manipulator path will have a specific structure as shown in Figure 2 and can be expressed as

$$\Pi^{bm} = \{(q_s^b, \pi_s^m), (q_1^b, \pi_1^m), ..., (q_n^b, \pi_n^m), (q_g^b, \pi_g^m)\}$$

We call this type of specific mobile manipulator path as an H-path. It consists of a set of mobile base poses q_i^b (blue dots) and for each base pose, there is a corresponding manipulator reconfiguration path π_j^m (white dots connected via light green edges). Each manipulator path π_j^m is a set of manipulator configurations. π_s^m, π_1^m, π_n^m, π_g^m are manipulator's paths at the mobile base states q_s^b, q_1^b, q_n^b, q_g^b respectively. Smaller ellipses have only one white dot, implies that manipulator path for those ellipses has only one configuration, in which case no reconfiguration motion of the manipulator is needed. It is implicit in the notation that the mobile manipulator motion from a node, say (q_i^b, π_i^m) to (q_j^b, π_j^m) consists of three steps: (i) at q_i^b, the manipulator will reconfigure by moving along π_i^m to the last configuration in π_i^m, (ii) with this new fixed manipulator configuration, the base moves to q_j^b, and (iii) manipulator again reconfigures by moving along π_j^m to the last configuration in π_j^m.

IV. THE HAMP ALGORITHM

We now describe the HAMP algorithm in detail explained in Algorithm 1.

In the first stage, a routine ConstructBaseRoadmap() is invoked to build a base roadmap in $C_{b_{free}}$ by randomly sampling base poses (with manipulator in a fixed home configuration) and connecting them as in BasicPRM routine in [24], and the pseudo-code for it is given in Algorithm 2. It constructs a base roadmap in an incremental manner until start and goal nodes are connected. Please note that while the sampling is in C_b, the collision checks are done for the entire mobile manipulator with manipulator in fixed home configuration. One could simply construct the base roadmap for base only, however, this could lead to several nodes/edges being invalidated in the subsequent stage. Algorithm 2 returns a connected base roadmap for start and goal base poses with manipulator in home configuration.

For second stage, we augment the node structure such that each node n, in addition to a base pose $n[q^b]$, and manipulator configuration $n[q^m]$, now has a best path field $n[p]$, a cost $n[c]$ (Euclidean metric in C_b) and a set of reconfiguration path fields $n[\text{RPATHS}^{[n_{adj}]}]$, one path for each adjacent node, n_{adj}.

The second stage described in Lines 5-25 of Algorithm 1 is a search mechanism that searches the base roadmap using a variant of Dijkstra's algorithm and SearchManipPath() routine in an intertwined manner. First, we change the manipulator configuration at start node, n_s, in the base roadmap from home q_H^m to a given configuration q_s^m and insert it in the search queue (Line 4). This is needed because the base roadmap was constructed with q_H^m which is different from q_s^m.

At each iteration of the while loop (Line 5), a node n is popped out from search queue and if the base component is not the base goal node then the manipulator configuration corresponding to node n is checked for collisions along each edge formed with adjacent nodes n' and in case a collision is detected, a reconfiguration path is searched for the manipulator. This is done by invoking a routine SearchManipPath(). If routine SearchManipPath() returns success as shown by the check on Line 17, then we insert adjacent node n' into the search queue and update the member variables at nodes n and n' (Lines 18-21). At node n, we update the reconfiguration path corresponding to adjacent node n', while at node n', we change the manipulator configuration with the last configuration q_{new}^m in the reconfiguration path and also update the new path and the corresponding cost. If the base component of popped out node $(n[q^b])$ from search queue is the base goal node $(n_g[q^b])$, then simply a manipulator reconfiguration path $\pi_{n_g}^m$ is searched from achieved manipulator configuration $\hat{q}_g^m$ to the desired manipulator configuration q_g^m

Algorithm 1: $\Pi^{bm} = \text{HAMP}(q_s, q_g, q_H^m)$

Input: $q_s := (q_s^b, q_s^m)$, $q_g := (q_g^b, q_g^m)$ and q_H^m
Output: H-path Π^{bm} from q_s to q_g

1 $G^b :=$ ConstructBaseRoadmap(q_s^b, q_g^b, q_H^m)
2 Augment node structure with best path $p := \emptyset$, cost $c := 0$ and reconfiguration paths Rpaths$^{[n_{adj}]} = \emptyset$ such that $n_i := \{q^b, q_H^m, p, c,$ Rpaths$^{[\cdot]}\}$
3 **while** ! TimeUp **do**
4 $Q \leftarrow n_s := \{q_s^b, q_s^m, \emptyset, 0, \emptyset\}$
5 **while** $Q \neq \emptyset$ ***and*** ! TimeUp **do**
6 $n :=$Pop(Q)
7 **if** $n[q^b] = n_g[q^b]$ **then**
8 $\pi_{n_g}^m =$SearchManipPathAtBaseGoal($\hat{q}_g^m, q_g^m$)
9 **if** $\pi_{n_g}^m = \emptyset$ **then**
10 Continue (go to step 5)
11 **end**
12 Exit (go to step 33)
13 **end**
14 **for** ***all*** n' of adj[n] ***and*** $n' \notin n[p]$ **do**
15 **if** $n[c]$+*cost*$[e_{nn'}] < n'[c]$ **then**
16 $(q_{new}^m, \pi_n^m) :=$SearchManipPath(n, n');
17 **if** $\pi_n^m \neq \emptyset$ **then**
18 $n'[c] := n[c]$+cost$[e_{nn'}]$
19 $n[$Rpaths$^{[n']}] := \pi_n^m$
20 $n' := \{-, q_{new}^m, n[p] \cup \{n'\}, n'[c], -\}$
21 $Q \leftarrow Q \cup \{n'\}$
22 **end**
23 **end**
24 **end**
25 **end**
26 **if** ! TimeUp **then**
27 **for** ***each*** *node* n_i *in* G^b **do**
28 $n_i := \{n_i[q^b], q_H^m, \emptyset, 0, \emptyset\}$
29 **end**
30 ExpandBaseRoadmap()
31 **end**
32 **end**
33 **return** Π^{bm}

at base goal pose (Lines 7-13). The final path is computed using $n_g[p]$ and $n_i[$Rpaths$^{[n_{adj}]}]$ such that $n_i, n_{adj} \in n_g[p]$.

If the search mechanism (stage 2) fails to find a path in the base roadmap constructed during stage 1, then we go back to stage 1 to further expand the base roadmap and the process repeats. The base roadmap expansion step is carried out from Lines 26-31 by invoking a routine ExpandBaseRoadmap() which randomly samples additional base poses and adds them to the base roadmap, as well as expands the base nodes in narrow regions. We have implemented the random-bounce walks strategy of standard PRM [9]. One could also use other strategies [24].

SearchManipPath() routine works as follows: it first checks if the manipulator configuration at base node n is collision-free along the edge formed with adjacent base node

Algorithm 2: $G^b = \text{ConstructBaseRoadmap}(q_s^b, q_g^b, q_H^m)$

1 $n_s :=$AddNode(q_s^b, q_H^m); $n_g :=$AddNode(q_g^b, q_H^m)
2 Create edge if CollisionFree(n_s, n_g)
3 **while** ! SameComponent(n_s, n_g) ***and*** ! TimeUp **do**
4 Sample base poses q_i^b from $C_{b_{free}}$ using a standard PRM sampling strategy to build base roadmap node set $\{n_i\}$ such that $n_i := (q_i^b, q_H^m)$
5 Create edge set $\{e_{ij}\}$ between nodes (n_i, n_j) if CollisionFree(n_i, n_j)
6 **end**
7 **return** $G^b = \{\{n_i\}, \{e_{ij}\}\}$

n' (Lines 1-4, Algorithm 3). If it is, then the returned manipulator path is that single configuration. This corresponds to the small purple ellipses with one white dot in Figure 2, which indicates that the manipulator configuration, corresponding to the white dot, is collision free along the edge and no manipulator planning was required. Otherwise, a set of manipulator configurations (goal^m)[2] are randomly sampled at base node n using a routine ComputeArmGoals(), described in Algorithm 4, such that each of the configuration in the goal^m is collision-free along the edge formed with an adjacent base node n'. The manipulator planning is carried out at base node n, by constructing an arm roadmap using an incremental PRM and a manipulator path is searched from manipulator configuration at base node n to any of the goal configurations in goal^m. To make the distinction between roadmap nodes in C_b and C_m , we use superscript m for the arm roadmap nodes in C_m.

We can divide the failures to solve the overall problem within permitted time in three types:

a) Type 1 - ConstructBaseRoadmap() fails to connect the start and goal configurations of the mobile manipulator with manipulator in home configuration for the entire base roadmap.
b) Type 2 - SearchManipPath() fails to search for a manipulator path, mainly because, either ComputeArmGoals() fails to find manipulator goal configurations within ArmGoalsTimeUp or SearchManipPath() fails to connect the start manipulator configuration at node n to any of the goal configurations reported by routine ComputeArmGoals() within ArmPlanningTimeUp.
c) Type 3 - path search reaches the goal node in the base roadmap but SearchManipPathAtBaseGoal() fails to compute a path from achieved manipulator configuration to the desired one.

V. RESULTS

We performed a series of simulations and real experiments on the SFU mobile manipulator roaming around in our lab to

[2]We are using multiple possible goal configurations because empirically it was faster than searching for a single goal configuration. Most likely, it is because the likelihood of multiple goal configurations being difficult to reach would be significantly lower.

Algorithm 3: $(q^m_{new}, \pi^m_n) = \text{SEARCHMANIPPATH}(n, n')$

Input: base roadmap nodes n, n' along an edge $e_{n,n'}$
Output: Manipulator path π^m_n at node n with q^m_n as start and q^m_{new} as goal such that q^m_{new} is collision-free along an edge $e_{n,n'}$

$n'' := (q^b_{n'}, q^m_n)$
if COLLISIONFREE(n, n'') **then**
  $q^m_{new} \leftarrow q^m_n$ and $\pi^m_n \leftarrow \{q^m_n\}$
end
else
  goal^m := COMPUTEARMGOALS$(n, n',$ KGoals$)$
  n^m_s :=ADDNODE(q^m_n); $n^m_{g_i}$:=ADDNODE(goal$^m[i]$)
  while ! ARMPLANNINGTIMEUP ***and*** ! TIMEUP **do**
    Sample q^m_i from $C_{m_{free}}$ using a standard PRM sampling strategy to build arm roadmap and search for a path π^m_n from n^m_s to $n^m_{g_i}$
    $q^m_{new} \leftarrow n^m_{g_i}$
  end
end
return (q^m_{new}, π^m_n)

Algorithm 4: goal^m = COMPUTEARMGOALS$(n, n',$ KGoals$)$

Input: base roadmap nodes n, n' and number of arm configurations (KGoals) to be computed
Output: a set of arm configurations as goals

while $i <$ KGoals ***and*** ! ARMGOALSTIMEUP **do**
  sample q^m_i from $C_{m_{free}}$
  $u \leftarrow \{q^b_n, q^m_i\}$ and $v \leftarrow \{q^b_{n'}, q^m_i\}$
  if COLLISIONFREE(u, v) **then**
    goal^m := goal$^m \cup \{q^m_i\}$; $i := i + 1$
  end
end
return goal^m

evaluate HAMP. We run our tests under linux on a Pentium dual core 2.5 Ghz computer with 4GB memory.

A. Simulation

We ran HAMP and a full 9D PRM on 5 different scenarios with varying levels of complexity in simulation and compared the outcomes. We used OMPL [12] to implement full 9D PRM in an incremental way (single-query) along with random-bounce walk expansion strategy [9] to connect narrow passages. The key parameters we used in HAMP are as follows: KGoals = 3, ARMGOALSTIMEUP = 2 seconds, and ARMPLANNINGTIMEUP = 6 seconds. For HAMP and full 9D PRM, we used 5 nearest neighbours to connect the new sample to the neighbouring nodes. Simulation environment corresponding to scenarios A and B is shown in Figure 1. The mobile manipulator with arm in vertically extended configuration was required to navigate from one side of the door to the other side. In A, the manipulator has no payload, whereas in B, we increased the complexity by adding a payload - a stick of 50cm length to the manipulator. In C, the task required passing a 50 cm long stick through a window of 40cm×50cm, while in D, the mobile manipulator needed to navigate through five cuboid obstacles in order to reach the goal. In scenario E, the mobile manipulator carrying a stick of 100 cm, enters from an open area into a very narrow corridor of width 80cm, navigates through the corridor and makes a turn through a very tight round corner and then finally exits into an open area, requiring frequent reconfiguration steps along the way.

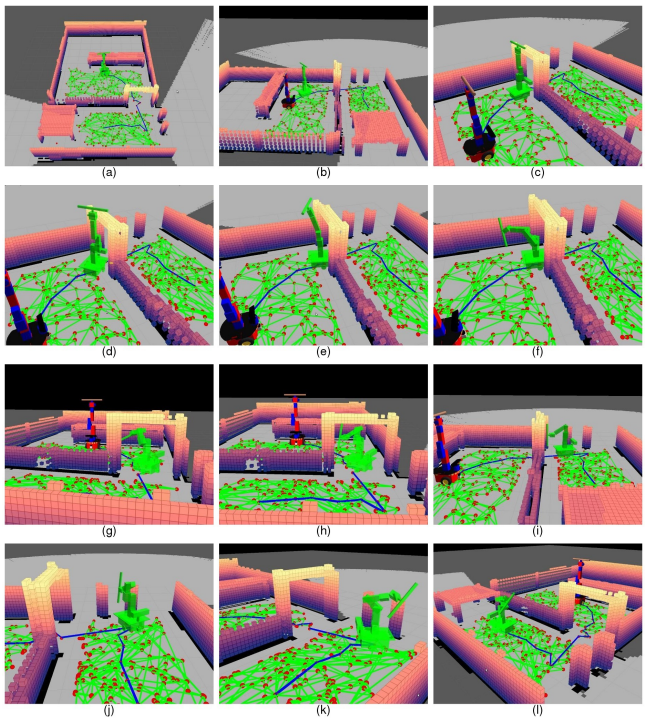

Fig. 3. Simulation test for scenario B: This simulation shows the mobile manipulator carrying a 50 cm stick as payload and navigating from one side of the door to the other side. The corresponding base roadmap and base path (blue color) of H-path are shown as well. (a) mobile manipulator's start configuration; (b), (c), (d) show a sequence of snapshots of mobile manipulator motions along the path with the same arm configuration; (e), (f) show the first reconfiguration step, in order to move along the path, arm configuration needs to be changed, as mobile manipulator can not cross the doorway due to arm and long payload collision with gate; (g), (h) show the second arm reconfiguration step; (i) the mobile manipulator advances along the path with arm in final configuration of the last reconfiguration step; (j), (k) show the third reconfiguration step; (l) mobile base has reached the goal but not the arm, this snapshot shows the mobile manipulator before the final reconfiguration step at goal.

Figure 3 shows snapshots of one of the simulation tests for scenario B. The manipulator changes its configuration 3 times along the path and once at the goal. Videos corresponding to the simulation tests for scenarios A, B, C, D are available online at `http://www.sfu.ca/~vpilania/research.html`, while for scenario E, the corresponding video is attached to this paper.

We compared HAMP and full 9D PRM on the basis of three criteria: (a) planner run time, (b) percentage of successful attempts, (c) base path length, and the results are presented in Table I. We observe from the table that for each of the scenarios A to E, full 9D PRM is outperformed by HAMP. HAMP solved the problems in less time (and less number of collision checks) with a higher success rate and

TABLE I

EXPERIMENTAL RESULTS IN 5 SCENARIOS FOR MOBILE MANIPULATOR PLANNING (HAMP VS. FULL 9D PRM).

	Permitted Time (s)	Planner	# Base Nodes		Total Arm Nodes		Time (s)		# Coll. Checks		#Reconfig /#Arm Checks.	B. Path Len. (m)	#Succ. /#Runs
			mean	s. d.	mean	s. d.	mean	s. d.	mean	s. d.			
A	40	HAMP	87	108	63	104	5.7	6.1	23k	25k	4/95	4.5	30/30
		PRM	489	326	N/A	N/A	12.5	10.7	80k	52k	N/A	6.1	24/30
B	40	HAMP	115	152	180	269	12.9	11.2	39k	36k	6/116	4.6	29/30
		PRM	823	267	N/A	N/A	19.5	14.0	130k	41k	N/A	7.8	7/30
C	120	HAMP	2	1	1958	920	36.9	19.4	173k	81k	1/1	0.8	30/30
		PRM	1638	265	N/A	N/A	84.2	21.3	297k	97k	N/A	4.9	12/30
D	70	HAMP	29	10	736	796	27.8	14.6	114k	76k	15/34	3.2	28/30
		PRM	945	491	N/A	N/A	38.2	18.2	192k	82k	N/A	4.3	21/30
E	300	HAMP	96	76	2897	1874	77.8	42.5	205k	134k	80/278	3.0	29/30
		PRM	10154	2469	N/A	N/A	193.5	48.9	786k	187k	N/A	4.7	23/30

shorter base path lengths as compared to full 9D PRM paths.

HAMP's failures for the scenarios B, D, E are of Type 2. In this case, the base roadmap gets denser and denser in an attempt to connect the narrow regions, and hence requires large number of calls to SEARCHMANIPPATH(). These failures happened because permitted time was over before the completion of calls to SEARCHMANIPPATH() for all the edges in the base roadmap. In E, sometimes (11 out of 30 runs) HAMP failed to find an H-path in the first iteration (Lines 1-25, Algorithm 1) even though the start and goal base poses were in the same connected component of the base roadmap. This is mainly because of the failure of manipulator planning (Type 2 and 3 failures). In that case, HAMP expands the base roadmap by adding more samples (lines 26-31, Algorithm 1) and successfully found the path in the subsequent iterations. On average, HAMP took 3 "expand roadmap" iterations to find a path for scenario E. In practice, it also attests to probabilistic completeness of HAMP, i.e., if it can not find a path in the first iteration then it adds more samples and repeats the search mechanism until a path is find or the permitted time to solve the problem is over. In addition, it is noteworthy that manipulator reconfiguration ("#Reconfig") is needed rather infrequently as compared to the number of times manipulator configurations were checked for collisions ("#Arm Checks") along the edges in the base roadmap as illustrated in the column "#Reconfig/#Arm Checks" in Table I. This supports our claim that HAMP moves the manipulator on a "need to" basis. Indeed the ratio is higher for more cluttered scenarios, but even for E, it is less than 30% (80/278).

Lastly, we illustrate the undesired motion of the arm in full 9D PRM vis a vis HAMP via a graphical representation, as shown in Figure 4, It shows a typical manipulator motion for HAMP and for full 9D PRM for scenarios B and E. Note that these motions were generated after a postprocessing step (path shortening) [25] . The figure shows the manipulator motion essentially as a histogram (red vertical lines) along the base path, shown in blue. For full 9D PRM (left figures), the height of each red line denotes the length of arm motion (in C_m) between two consecutive poses along the discretized base path. For HAMP, the manipulator motion takes place at fixed base configurations, hence the length is zero except at some base poses, where the height of red line denotes the length of the manipulator reconfiguration path (right figures).

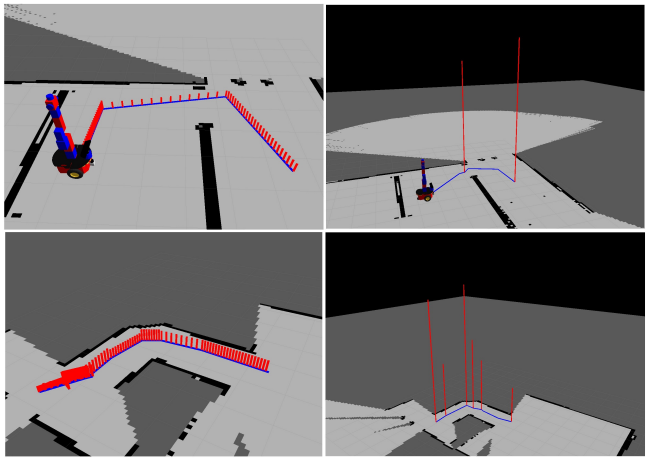

Fig. 4. This figure shows a comparison of manipulator motion (the height of red lines corresponds to the length of arm motion (in C_m)) along the base path (blue line) between full 9D PRM (left) and HAMP (right) for scenarios B (top) and E (bottom). Please see text for explanation.

It clearly supports our claim that HAMP avoids undesired arm motions as the base moves along a path which is not the case with full 9D PRM.

B. Preliminary Experiments on SFU Mobile Manipulator

We also tested HAMP on SFU mobile manipulator in our lab. Our experimental set-up consists of two doorways, 2nd doorway is just 10 cm above the minimum height that can be attained by the manipulator. We carried out 18 experiments on SFU mobile manipulator with the experimental setup, the minimum and maximum planner runtime was noted to be 2.43 seconds and 7.8 seconds, respectively. Video corresponding to real experiment on SFU mobile manipulator is available online at `http://www.sfu.ca/~vpilania/research.html`. In real experiments, for now, we are not able to demonstrate the pole example due to lack of a gripper in SFU mobile manipulator.

VI. CONCLUSION AND FUTURE WORK

We presented a hierarchical and adaptive mobile manipulator planner (HAMP) which searches in two sub-spaces, the base sub-space and the manipulator sub-space, in a novel way and on a "need to" basis, i.e., the search in manipulator space is invoked only for those points in the base-space where it is needed. We proved that HAMP is probabilistically complete. We implemented HAMP for a 9 degree of freedom

mobile manipulator and showed that it can find a path in environments where conservative approaches will fail since manipulator configuration needs to be changed several times while navigating from start to goal. We also compared HAMP with full 9D PRM and observed that HAMP takes less time to compute the paths with a higher success rate. In addition, the base path length corresponding to the H-path given by HAMP is significantly less than the length of base path given by the full 9D PRM. HAMP also avoids the undesired arm motions as the base moves along a path.

Our expectations are that the benefits of HAMP will be similar with RRT (or Bi-directional RRT) as core underlying sub-planners. We would like to implement such a planner and empirically show this. We would also like to incorporate task space constraints within the HAMP framework, e.g., by substituting SEARCHMANIPPATH() with ATACE [26] or CBiRRT [27] or [28] that does incorporate task constraints. While we have focussed on a specific platform, i.e., mobile manipulator, where the hierarchy is defined across the base and the manipulator, the HAMP approach can be generalized over any hierarchical abstraction over the entire configuration space. We intend to explore this.

ACKNOWLEDGMENT

This research is supported, in part, by an NSERC Discovery Grant to Kamal Gupta. We would like to thank the anonymous referees for their insightful comments that helped us improve the paper.

REFERENCES

[1] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Sucan, "Towards reliable grasping and manipulation in household environments," in *RSS 2010 Workshop on Strategies and Evaluation for Mobile Manipulation in Household Environments*, 2010, pp. 1–12.

[2] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[3] A. Monastero and P. Fiorini, "Target pose computation for non-holonomic mobile manipulators," in *Proc.of the IEEE International Conference on Advanced Robotics*, 2009, pp. 1–8.

[4] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments," in *Proc.of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 1–6.

[5] D. Berenson, J. Kuffner, and H. Choset, "An optimization approach to planning for mobile manipulation," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2008, pp. 1187–1192.

[6] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[7] J. Scholz, S. Chitta, B. Marthi, and M. Likhachev, "Cart pushing with a mobile manipulation system: Towards navigation with moveable objects," in *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[8] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. 98-11, 1998.

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

[10] J. Barraquand, B. Langlois, and J. C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[11] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 995–1001.

[12] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, 2012. [Online]. Available: http://ompl.kavrakilab.org

[13] H. G. Tanner and K. J. Kyriakopoulos, "Nonholonomic motion planning for mobile manipulators," in *Proc.of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000, pp. 1233–1238.

[14] J. Tan and N. Xi, "Unified model approach for planning and control of mobile manipulators," in *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001, pp. 3145–3152.

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

[16] Y. Yang and O. Brock, "Elastic roadmaps - motion generation for autonomous mobile manipulation," *Autonomous Robot*, vol. 28, no. 1, pp. 113–130, 2010.

[17] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in three-dimensional cluttered environments for mobile manipulation," in *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2012.

[18] A. Hornung and M. Bennewitz, "Adaptive level-of-detail planning for efficient humanoid navigation," in *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[19] J. Yang, P. Dymond, and M. Jenkin, "Exploiting hierarchical probabilistic motion planning for robot reachable workspace estimation," in *Informatics in Control Automation and Robotics, LNEE85*. Springer-Verlag, 2011, pp. 229–241.

[20] K. Gochev, A. Safonova, and M. Likhachev, "Planning with adaptive dimensionality for mobile manipulation," in *Proc.of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2012, pp. 2944–2951.

[21] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes," *IEEE Transctions on Robotics*, vol. 24, no. 5, 2008.

[22] Y. Yu and K. Gupta, "C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments," *International Journal of Robotics Research*, vol. 23, no. 12, pp. 1197–1223, 2004.

[23] L. Torabi, M. Kazemi, and K. Gupta, "Configuration space based efficient view planning and exploration with occupancy grids," in *Proc.of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 2827–2832.

[24] D. Hsu, J. C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *International Journal of Robotics Research (IJRR)*, vol. 25, no. 7, pp. 627–643, 2006.

[25] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005, ch. 7, pp. 212–214.

[26] Z. Yao and K. Gupta, "Path planning with general end-effector constraints," *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 316–327, 2007.

[27] D. Berenson, S. Srinivasa, D. Ferguson , and J. Kuffner, "Manipulation planning on constraint manifolds," in *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 625–632.

[28] M. Stilman, "Task constrained motion planning in robot joint space," in *Proc.of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 3074–3081.