Autonomous Robots manuscript No.
(will be inserted by the editor)

A hierarchical and adaptive mobile manipulator planner with base pose uncertainty

Vinay Pilania · Kamal Gupta

Received: date / Accepted: date

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 and a path from current configuration to the new reachable configuration is computed. We show that HAMP is probabilistically complete. We compared HAMP with full 9D PRM and observed that the full 9D PRM is outperformed by HAMP 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. We also evaluated the tree versions of HAMP, with RRT and Bi-directional RRT as core underlying sub-planners, and observed similar advantages, although the time saving for Bi-directional RRT version is modest. We then present an extension of HAMP (we call it HAMP-U) that uses belief space planning to account for localization uncertainty associated with the mobile base position and ensures that the resultant path for the mobile manipulator has low uncertainty at the goal. Our experimental results show that the paths generated by HAMP-U are less likely to result in collision and are safer to execute than those generated by HAMP (without incorporating uncertainty), thereby showing the importance of incorporating base pose uncertainty in our overall HAMP algorithm.

Keywords Mobile manipulation · Planning under uncertainty · Motion planning · Navigation · Adaptive

Vinay Pilania · Kamal Gupta
Robotic Algorithms & Motion Planning (RAMP) Lab
School of Engineering Science, Simon Fraser University
Burnaby, BC V5A1S6, Canada
E-mail: vpilania@sfu.ca

Kamal Gupta
E-mail: kamal@sfu.ca

1 Introduction

Autonomous mobile manipulation includes several intertwined sub-problems including finding suitable grasps and grasping the object (Ciocarlie et al, 2010; Saxena et al, 2008), finding the best mobile base location and manipulator configuration corresponding to the end effector pose (Monastero and Fiorini, 2009), close-range scene segmentation for table-top manipulation (Rusu et al, 2009) with static mobile base, and finding the mobile manipulator path from start to goal (base poses and manipulator configurations) (Berenson et al, 2008). 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 (Berenson et al, 2008; Marder-Eppstein et al, 2010; Scholz et al, 2011) 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.

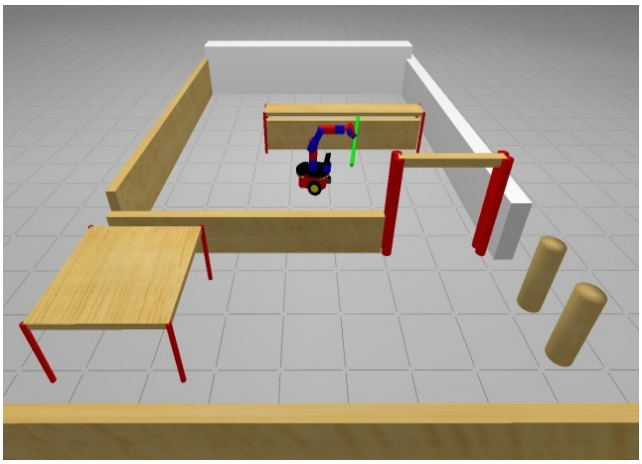


Fig. 1 This example (simulation environment for scenarios A and B in Section 7) shows the mobile manipulator carrying a long payload (120 cm long stick) and it needs to pass through a doorway to reach the goal on the other side.

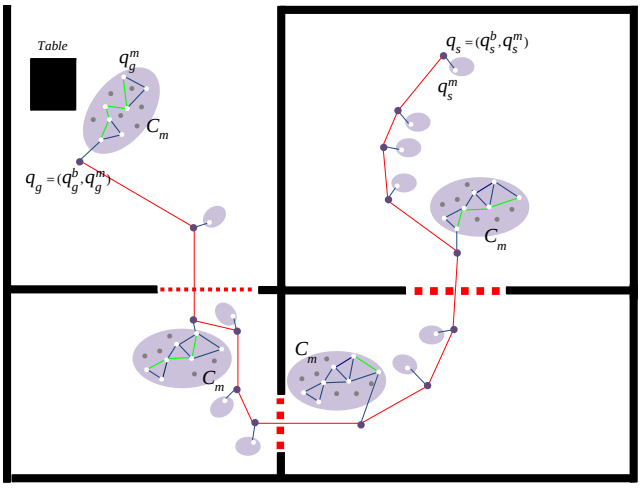


Fig. 2 A schematic illustrating the planned mobile manipulator path Π^{bm} given by HAMP algorithm. Please see text for explanation.

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.

One possible solution to this motion planning problem is to use sampling based planners (Lavalle, 1998; Kavraki et al, 1996) in full configuration (C-space) of the mobile manipulator. However, besides being somewhat computationally expensive, the computed path for the mobile manipulator may result into 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. Furthermore, it is generally difficult to ensure tight error bounds on the mobile base that are comparable to those for the arm and hence synchronizing controllers between the two can be difficult. Therefore, it is quite reasonable to execute arm and base motions sequentially, and within this overall paradigm, our HAMP approach, as outlined below, is quite reasonable[1].

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 Fig 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 home[2] 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 manipula-

[1] Thanks to the anonymous referee for pointing this out.

[2] 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.

tor 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.

Fig 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 Sec 4), 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. Our choice to use PRM as underlying core sub-planner (for base and for the manipulator) within the HAMP framework is primarily because when we incorporate base uncertainty (as explained in the next paragraph) in the mobile manipulator paths, it allows us to optimize the paths with respect to the base uncertainty (at the goal). This would not be the case if we were to use tree versions of sampling based planners (such as RRT (Lavalle, 1998)) as core sub-planners within HAMP. However, RRT (in the absence of uncertainty)

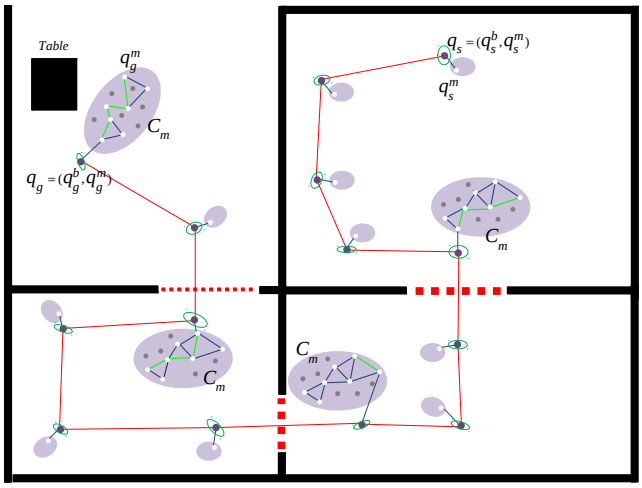


Fig. 3 A schematic illustrating the planned mobile manipulator path Π^{bm} given by HAMP-U algorithm. Please see text for explanation.

is generally more efficient in terms of planning time than PRM, especially the Bi-directional RRT (Kuffner and LaValle, 2000). Therefore, we also evaluated the tree versions of HAMP with RRT and Bi-directional RRT as the core underlying sub-planners.

Most traditional motion planning algorithms assume deterministic motion and leave the issues of uncertainty to the control phase in which the path is executed with a feedback controller. However, a mobile base inherently has localization uncertainty due to wheel slippage and other unmodeled errors. It is important to consider this uncertainty in the planning stage for safe and collision-free execution of motion plans. Hence, we extend our HAMP approach - we call it HAMP-U - to account for localization uncertainty associated with the mobile base position and a schematic illustrating the planned mobile manipulator path is given in Fig 3. Blue dots correspond to mean base pose nodes, and the uncertainty in position is shown by ellipses (green color). In HAMP-U, in the first stage, the base roadmap is substituted by a belief roadmap (BRM) in the belief-space of mobile base, using the approach proposed by Prentice and Roy (2009) with additional modifications explained later in Sec 6. In the second stage, the search algorithm searches for the mobile manipulator path in the BRM by propagating base pose uncertainty from start to goal, in a manner that minimizes the goal uncertainty, as in (Prentice and Roy, 2009), again, with some modifications. Note that, in Fig 3, the mobile base path detours from the shortest path (Fig 2) through sensing-rich environment to remain well-localized, a direct and well known consequence of standard BRM. Due to this low uncertainty in base position, the planned mobile manipulator motions are also less likely to result in collision as validated in our experiments. While mobile robotics

literature (mobile base only) has extensively considered uncertainty (world is 2-dimensional in most of these cases, although some recent work has considered 3D world, but for SLAM and not planning), to the best of our knowledge, this uncertainty is largely ignored in mobile manipulation. Huang and Gupta (2009) considered this, but for the case of fixed mobile base only. Our framework in HAMP-U considers this uncertainty explicitly by embedding BRM within the overall HAMP algorithm.

In summary, the key contributions of our research work are: 1) a Hierarchical and Adaptive Mobile Manipulator Planner (HAMP) that moves the arm in a judicious and on a "need to" basis; 2) a mathematical proof to show that HAMP is probabilistically complete; 3) an algorithm HAMP-U (uncertainty extension to HAMP) that extends the HAMP framework while accounting for localization uncertainty associated with mobile base position.

2 Related work

In this section we review the related work and place our research work into context. First, we review the related work on mobile manipulation planning, and then we consider the work concerning motion planning under uncertainty.

2.1 Mobile manipulation planning

Most of the previous work (Tanner and Kyriakopoulos, 2000; Tan and Xi, 2001; Yamamoto and Yun, 1994; Yang and Brock, 2010) 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, Marder-Eppstein et al (2010) and Scholz et al (2011) use a compact 3D representation of the environment, but path planning is accomplished in a projected 2D environment representation with a 2D footprint of the mobile manipulator. Such an approach will fail, for example, where the mobile manipulator is required to push and store a cart under a table. Hornung et al (2012) improved upon (Marder-Eppstein et al, 2010; Scholz et al, 2011) 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 (2012) 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 (Yang et al, 2011). An interesting use of adaptive dimensionality has recently been introduced in Gochev et al (2012). 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 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 real time in dynamic environments is presented in Vannoy and Xiao (2008). The planner takes advantage of redundancy in optimizing overall motion via randomly invoking a "stop" genetic operator that allows for either the base or the manipulator to remain stationary during a portion of the trajectory. 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.

2.2 Planning under uncertainty

While standard motion planning algorithms often assume that a mobile base can track its position reliably during path execution stage (as is the case with HAMP), in reality, there is always some uncertainty associated with mobile base position. The uncertainty typically originates from three sources: (i) motion uncertainty - uncertainty in a robot's motion often caused by factors such as wheel slippage, (ii) sensor uncertainty - uncertainty in its sensory readings, and (iii) map uncertainty - uncertainty in the environment map or imperfect locations of features (information sources) in the environment. Our key motivation to extend HAMP to consider this uncertainty in the planning stage is mainly to ensure safe and collision-free execution of motion plans. The uncertainty associated with the mobile base is typically of the order of few tens of centimeters and there are no significant dynamics associated with the motion of the base (unlike for exam-

ple, an aerial vehicle). Hence, in our case, extensions of the sampling based framework to incorporate uncertainty, e.g., the belief roadmap (BRM) approach of Prentice and Roy (2009), are more appropriate. These approaches, essentially add an uncertainty dimension(s) to the robot state and each belief state then is a combination of robot state and the associated uncertainty (Huang and Gupta, 2008; Melchior and Simmons, 2007; Gonzalez and Stentz, 2005; Lambert and Gruyer, 2003; Roy and Thrun, 1999). Furthermore, we assume that the controller is capable of driving the state estimate back to the desired path, a reasonable assumption for slow moving vehicles such as the mobile base in our mobile manipulator. With these characteristics in mind, the belief space roadmap (BRM) approach was incorporated within our HAMP Framework. An attractive aspect of BRM is that, while it explicitly simulates measurements along candidate paths and then chooses the path with minimal uncertainty at the goal, it uses covariance factorization techniques to significantly reduce the computation burden of this process.

More recent approaches have also accounted for the controller in the planning stage, e.g., Platt et al (2010) and van den Berg et al (2011), however, there is significant increase in the computational cost. Nevertheless, these could also be incorporated within the HAMP framework. We also mention that partially observable Markov decision process (POMDP) (Smallwood and Sondik, 1973; Kaelbling et al, 1998) is a general framework to deal with motion and sensing uncertainty, however due to its significant complexity, solving realistic problems with large state spaces remains a challenge, even though progress has been made on the efficiency issues of these approaches (Pineau et al, 2003; Kurniawati et al, 2009, 2012). Kurniawati et al (2012) also takes into account mapping uncertainty by embedding Guibas et al (2008) within POMDP framework. A recent framework, SLQG-FIRM, proposed by Aghamohammadi et al (2014) extends the sampling based framework to belief space. Finally, Guibas et al (2008); Missiuro and Roy (2006); Burns and Brock (2007); Nakhaei and Lamiraux (2008) consider the mapping uncertainty about the environment but not the motion and sensing uncertainty. Another class of planners Bouilly et al (1995); Lazanas and Latombe (1995); Fraichard and Mermond (1998) assumes the presence of landmark regions in the environment where accumulated motion uncertainty can be "reset".

3 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 Yu and Gupta (2004); Torabi et al (2007).

Given a 3D map, the start $q_s = (q_s^b, q_s^m)$ and goal $q_g = (q_g^b, q_g^m)$ configurations of the mobile manipulator, 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 Fig 2 and can be expressed as

$$\Pi^{bm} = \{(q_s^b, \pi_s^m), (q_{r_2}^b, \pi_2^m), ..., (q_{r_{n-1}}^b, \pi_{n-1}^m), (q_g^b, \pi_g^m)\}$$

We call this type of specific mobile manipulator path as an H-path (short for HAMP-path). It consists of a sequence of poses $q_{r_i}^b$ at which the manipulator reconfiguration step takes place (subscript r denotes reconfiguration), i.e., the base remains stationary and the manipulator moves along path π_i^m to a new configuration, the end point of π_i^m. It is implicit in the notation that the mobile manipulator motion from a node, say $(q_{r_i}^b, \pi_i^m)$ to $(q_{r_{i+1}}^b, \pi_{i+1}^m)$ consists of three steps: (i) at $q_{r_i}^b$, 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 along base path segment π_i^b to $q_{r_{i+1}}^b$, and (iii) manipulator again reconfigures by moving along π_{i+1}^m to the last configuration in π_{i+1}^m.

4 The HAMP algorithm

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

In the first stage, ConstructBaseRoadmap() routine is invoked to build a base roadmap in $C_{b_{free}}$ by randomly sampling base poses (with manipulator in a fixed

Algorithm 1: Π^{bm} = 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 $\text{RPATHS}^{[n_{adj}]} = \emptyset$ such that $n_i := \{q^b, q_H^m, p, c, \text{RPATHS}^{[\cdot]}\}$
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, π_n^m) :=SearchManipPath(n, n')
17 **if** $\pi_n^m \neq \emptyset$ **then**
18 $n'[c] := n[c]$+cost$[e_{nn'}]$
19 $n[\text{RPATHS}^{[n']}] := \pi_n^m$
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}

Algorithm 2: G^b=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** $G^b = \{\{n_i\}, \{e_{ij}\}\}$

home configuration) and connecting them as in BasicPRM routine in (Hsu et al, 2006), and the pseudocode 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 at base goal pose (Lines 7-13). The final path is computed using $n_g[p]$ and $n_i[\text{RPATHS}^{[n_{adj}]}]$ 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

Algorithm 3: (q^m_{new}, π^m_n)=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 $e_{n,n'}$

1 $n'' := (q^b_{n'}, q^m_n)$
2 **if** CollisionFree(n, n'') **then**
3 $\quad q^m_{new} \leftarrow q^m_n$ and $\pi^m_n \leftarrow \{q^m_n\}$
4 **end**
5 **else**
6 $\quad$ goal^m := ComputeArmGoals$(n, n'$, KGoals$)$
7 $\quad n^m_s$:=AddNode(q^m_n); $n^m_{g_i}$:=AddNode(goal$^m[i]$)
8 $\quad$ **while** ! ArmPlanningTimeUp ***and*** ! TimeUp **do**
9 $\quad\quad$ 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 $n^m_{g_i}$
10 $\quad\quad q^m_{new} \leftarrow n^m_{g_i}$
11 $\quad$ **end**
12 **end**
13 **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

1 **while** $i <$ KGoals ***and*** ! ArmGoalsTimeUp **do**
2 $\quad$ sample q^m_i from $C_{m_{free}}$
3 $\quad u \leftarrow \{q^b_n, q^m_i\}$ and $v \leftarrow \{q^b_{n'}, q^m_i\}$
4 $\quad$ **if** CollisionFree(u, v) **then**
5 $\quad\quad$ goal^m := goal$^m \cup \{q^m_i\}$; $i := i + 1$
6 $\quad$ **end**
7 **end**
8 **return** goal^m

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 (Kavraki et al, 1996). One could also use other strategies (Hsu et al, 2006). Please note that this expansion of the base roadmap (possibly repeated multiple times), allows HAMP to deal with narrow passages as well (theoretically, we show HAMP is probabilistically complete). Let's say we pop out a node in the base roadmap near the entrance to a narrow passage. It is not possible to get into the narrow passage without reconfiguring the arm. However, we are so close to the entrance that the arm can not be reconfigured into the proper configuration for entry (it would hit the walls). In this case, HAMP will pop out the next node from the search queue and will try to enter the narrow passage through different base paths. For instance, it might try a base node away from the entrance, reconfigure the arm such at this node, that might allow it to enter the narrow passage (e.g., see Scenario E, Figure 7 (c) in Section 7.2).

Now we explain the SearchManipPath() routine which 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 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 Fig 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)[3] 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. The current version of SearchManipPath() does not incorporate any task space constraints (such as keeping the payload orientation vertical), however in the future versions, this will be replaced by the *ATACE* (Alternate Task and Configuration Space) algorithm (Yao and Gupta, 2007; Stilman, 2007) that does incorporate task constraints.

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.

[3] 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.

5 Probabilistic completeness proof

This section deals with probabilistic completeness proof of HAMP. Suppose q_s, $q_g \in C_{bm_{free}}$ are two mobile manipulator configurations that can be connected by an H-path in $C_{bm_{free}}$. HAMP is considered to be probabilistic complete, if for any given (q_s, q_g)

$$\lim_{\substack{n\to\infty \\ m\to\infty}} Pr[(q_s, q_g)FAILURE] = 0 \tag{1}$$

where $Pr[(q_s, q_g)FAILURE]$ denotes the probability that HAMP fails to answer the query (q_s, q_g) after a base roadmap in $C_{b_{free}}$ with n samples and manipulator roadmaps each with m samples in $C_{m_{free}}$ have been constructed. The outline of the probabilistic completeness proof is as follows: First we assume that an H-path Π^{bm} from q_s to q_g exists - recall that an H-path consists of a sequence of sub-paths where the base moves with fixed manipulator configuration (base path segments), followed by a reconfiguration step where base is fixed but manipulator moves to a different configuration (re-configuration path). We then tile the base path as well as all the manipulator paths with a set of carefully chosen balls such that generating a sample in each ball ensures that these samples can be connected with appropriate collision-free edges and hence a collision-free H-path, $\hat{\Pi}^{bm}$ between q_s and q_g will be found by HAMP and the probability of generating such samples approaches 1 with increasing m and n.

Assume an H-path Π^{bm} from q_s to q_g with k (finite but can be arbitrarily large) manipulator reconfiguration steps exists - the path is composed of k base path segments as shown in Fig 4. For each base path segment π^b_i, the mobile manipulator moves with a fixed manipulator configuration; at the end of the segment, denoted by base pose $q^b_{r_i}$, a manipulator reconfiguration step is executed (with base stationary), and the mobile manipulator now moves along the next base path segment π^b_{i+1} with the new fixed manipulator configuration. Let $Q^b = \{q^b_{r_i}\}$ denote the set of base poses along π^b at which reconfiguration steps take place. Let the length of the entire base path be L^b and the length of manipulator path for i^{th} reconfiguration step be $L^m_{r_i}$. Lastly, let d_b, d_m denote the dimensions of C_b and C_m, respectively.

We now define three clearances. The clearance of π^b_i, denoted $\rho_i = clr(\pi^b_i)$, is the farthest distance (in C_b) away from the path segment at which a given base pose with manipulator in the fixed configuration (provided by Π^{bm}) can be guaranteed to be collision-free. If π^b_i lies in $C_{b_{free}}$, then $clr(\pi^b_i) > 0$. Let $\rho_b = clr(\pi^b) = min_i(\rho_i)$ be the clearance along the entire base path π^b. The clearance of the base pose $q^b_{r_i} \in Q^b$, denoted

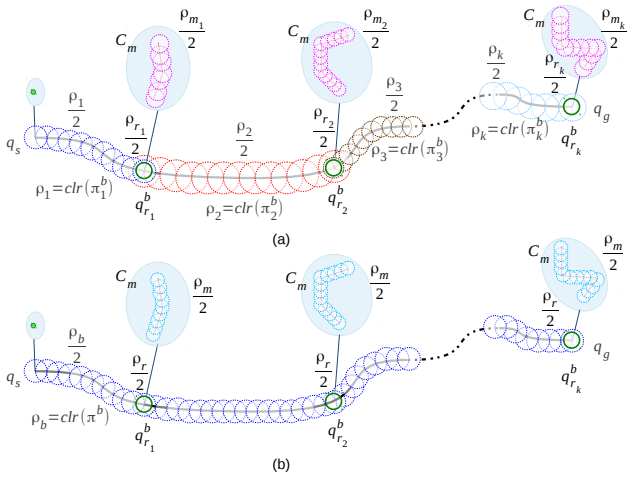


Fig. 4 A schematic illustrating a known mobile manipulator path and the tiling with carefully chosen balls. (a) clearance is not uniform through out the base path and for all the manipulator paths, for e.g., for a clearance of ρ_i the corresponding base path segment is tiled with balls each of radius $\frac{\rho_i}{2}$; (b) clearance is the minimum of all balls in (a), we tile the base path π_b with balls each of radius $\frac{\rho_b}{2}$, base poses $q^b_{r_i} \in Q^b$ with balls of radius $\frac{\rho_r}{2}$, and the manipulator reconfiguration paths with balls of radius $\frac{\rho_m}{2}$.

$\rho_{r_i} = clr(q^b_{r_i})$, is the farthest distance (in C_b) from $q^b_{r_i}$ at which the manipulator reconfiguration path (provided by Π^{bm}) can be executed collision-free. Again, let $\rho_r = min_i(\rho_{r_i})$. The clearance of a manipulator path corresponding to a base pose $q^b_{r_i}$ (tiled with a ball of radius $\frac{\rho_{r_i}}{2}$), denoted ρ_{m_i}, is defined (in C_m) as the minimum of all the clearances that can be obtained for a manipulator path corresponding to any sampled base pose from the ball. Let $\rho_m = min_i(\rho_{m_i})$.

The measure μ denotes the volume of a region of space, e.g., $\mu(B_\epsilon(x))$ measures the volume of an open ball $B_\epsilon(x)$ of radius ϵ centered at x. If $A \subset C_{b_{free}}$ is a measurable subset and x is a random point chosen from $C_{b_{free}}$ by sampling strategy of standard PRM, then

$$Pr(x \in A) = \frac{\mu(A)}{\mu(C_{b_{free}})} \tag{2}$$

We now tile the base path π_b with balls each of radius $\frac{\rho_b}{2}$, base poses $q^b_{r_i} \in Q^b$ with balls of radius $\frac{\rho_r}{2}$ (green circles), and the manipulator reconfiguration paths with balls of radius $\frac{\rho_m}{2}$ as shown in Fig 4 (b). Let $p^b = \lceil \frac{2L^b}{\rho_b} \rceil$ and observe that there are p^b points (centers of balls) on the path π^b such that $dist^b(q^b_i, q^b_{i+1}) < \rho_b$, where $dist^b$ is a Euclidean metric on $\mathbb{R}^{d_b}$. Out of these p^b points, there are k points (Q^b) where a manipulator reconfiguration step is needed. Let $\neg Q^b$ denote the set of remaining $(p^b - k)$ points along π^b. Let $y_i \in B_{\rho_b/2}(q^b_i)$ and $y_{i+1} \in B_{\rho_b/2}(q^b_{i+1})$. Then the line segment $\overline{y_i y_{i+1}}$ must lie inside $C_{b_{free}}$ since both endpoints lie in the ball

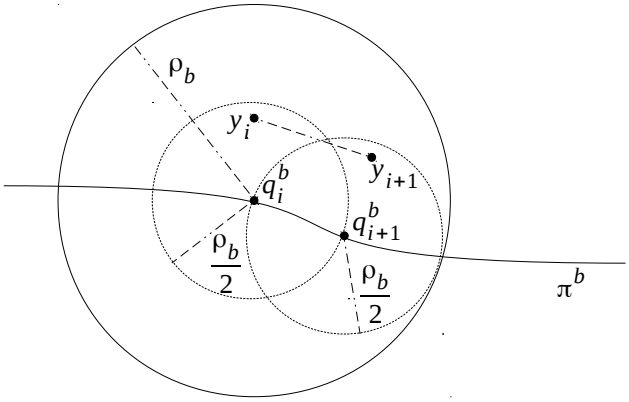


Fig. 5 A closer look at base path tiling with balls of radius $\frac{\rho_b}{2}$. Points y_i and y_{i+1} are inside the balls and therefore, the line segment $\overline{y_i y_{i+1}}$ must lie inside $C_{b_{free}}$ since both endpoints lie in the ball $B_{\rho_b}(q_i^b)$.

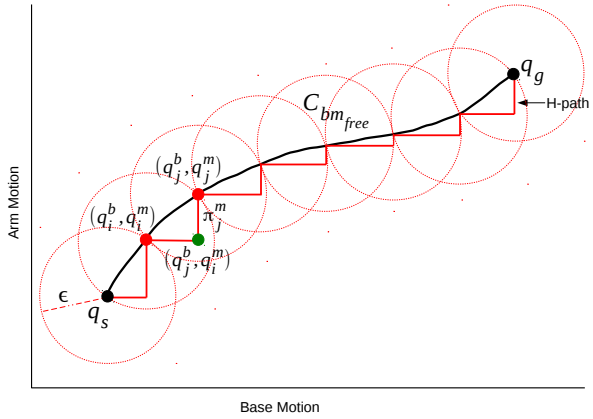


Fig. 6 This figure shows that any path in $C_{bm_{free}}$ (assuming it is an open set) connecting the start (q_s) and the goal (q_g) configurations can be approximated by an H-path, also lying in $C_{bm_{free}}$. The black curve denotes an arbitrary path that lies in $C_{bm_{free}}$. Let ϵ be minimum clearance (from C-obstacles) along the path (it must exist since $C_{bm_{free}}$ is open). The path is then tiled with balls of radius ϵ. The H-path approximation is shown in red and by construction, it is guaranteed to be collision-free.

$B_{\rho_b}(q_i^b)$ as shown in Fig 5. Let $p_{r_i}^m = \lceil \frac{2L_{r_i}^m}{\rho_m} \rceil$ and observe that there are $p_{r_i}^m$ points on the i^{th} manipulator reconfiguration path such that $dist^m(q_i^m, q_{i+1}^m) < \rho_m$, where $dist^m$ is a Euclidean metric on $\mathbb{R}^{\mathrm{d_m}}$.

Let $V^b \subset C_{b_{free}}$ be a set of n base poses generated uniformly at random by HAMP for the construction of base roadmap. Similarly, $V_{r_i}^m \subset C_{m_{free}}$ be a set of m manipulator configurations generated uniformly at random by HAMP for the construction of i^{th} manipulator roadmap. If the following conditions hold then an H-path from q_s to q_g will be found: (a) each ball along the base path π^b corresponding to $q_i^b \in \neg Q^b$ gets atleast one sample, i.e., there is a subset $\{y_i\} \subset V^b$ of $(p^b - k)$ base poses such that $y_i \in B_{\rho_b/2}(q_i^b)$; (b) remaining balls along π^b corresponding to $q_{r_i}^b \in Q^b$ get atleast one sample each, i.e., subset $\{y_i'\} \subset V^b$ of k base poses such that $y_i' \in B_{\rho_r/2}(q_{r_i}^b)$; (c) for all the manipulator paths, the corresponding balls also get atleast one sample each, i.e., there is a subset $\{y_i^r\} \subset V_{r_i}^m$ of $p_{r_i}^m$ manipulator configurations such that $y_i^r \in B_{\rho_m/2}(q_i^m)$. As mentioned earlier, these conditions ensure at least one sample in each ball such that these samples can be connected with appropriate collision-free edges and hence a collision-free H-path will be found by HAMP.

To formalize it mathematically, let $I_1, ..., I_{p^b}$ be a set of indicator variables such that each I_j (excluding those k indicators where manipulator reconfiguration is needed) witness the event that there is a $y \in V^b$ and $y \in B_{\rho_b/2}(q_i^b)$ while remaining k indicator variables witness the event that there is a $y' \in V^b$ and $y' \in B_{\rho_r/2}(q_{r_i}^b)$. Let $I_{r_1}, ..., I_{r_k}$ be a set of indicator variables such that each I_{r_i} witness the event that all balls along the i^{th} manipulator reconfiguration path get at least one sample each. Let $I_1^m, ..., I_{p_{r_i}^m}^m$ be a set of indicator variables for each I_{r_i} such that $I_{r_i} = I_1^m \wedge I_2^m \wedge, ..., \wedge I_{p_{r_i}^m}^m$ and each I_t^m witness the event that there is a $y^r \in V_{r_i}^m$ and $y^r \in B_{\rho_m/2}(q_i^m)$. It follows that HAMP succeeds in answering the query (q_s, q_g) if $I_j = 1$ for all $1 \leq j \leq p^b$ and $I_{r_i} = 1$ for all $1 \leq i \leq k$. If at least one of the indicator variables (I_j, I_{r_i}) is 0 then HAMP would fail. Therefore, the probability of failure (Equation 1) then can be written as

$$Pr[(q_s, q_g)FAILURE] \leq Pr((\bigvee_{j=1}^{p^b} I_j = 0) \vee (\bigvee_{i=1}^{k} I_{r_i} = 0)) \tag{3}$$

$$\leq \sum_{j=1}^{p^b} Pr[I_j = 0] + \sum_{i=1}^{k} Pr[I_{r_i} = 0] \tag{4}$$

where the last inequality follows from the union bound. We further breakdown the first term into two components: (i) $(p_b - k)$ balls along π^b where manipulator reconfiguration is not needed, and (ii) remaining k balls each with a radius of $\frac{\rho_r}{2}$ where reconfiguration is needed. Therefore, RHS of last inequality (Equation 4) can be written as

$$\leq \sum_{j \in \neg Q^b} Pr[I_j = 0] + \sum_{j \in Q^b} Pr[I_j = 0] + \sum_{i=1}^{k} Pr[I_{r_i} = 0] \tag{5}$$

The probability of a given $I_j = 0$ is computed by observing that none of the n randomly generated indepen-

dent samples falls in $B_{\rho_b/2}(q_j^b)$, therefore for $j \in \neg Q^b$,

$$Pr[I_j = 0] = \left(1 - \frac{\mu(B_{\rho_b/2}(q_j^b))}{\mu(C_{b_{free}})}\right)^n \tag{6}$$

However,

$$\frac{\mu(B_{\rho_b/2}(\cdot))}{\mu(C_{b_{free}})} = \frac{(\frac{\rho_b}{2})^{d_b}\mu(B_1(\cdot))}{\mu(C_{b_{free}})} = \sigma_1 \rho_b^{d_b} \tag{7}$$

where $B_1(\cdot)$ is the unit ball in $\mathbb{R}^{\mathrm{d_b}}$ and $\sigma_1 = \mu(B_1(\cdot))/2^{d_b}\mu(C_{b_{free}})$. Hence RHS of Equation 5 becomes

$$\leq (p^b - k)(1-\sigma_1\rho_b^{d_b})^n + k(1-\sigma_1\rho_r^{d_b})^n + \sum_{i=1}^{k} Pr[I_{r_i} = 0] \tag{8}$$

$I_{r_i} = 0$ only if $\vee_{t=1}^{p_{r_i}^m} I_t^m = 0$. Therefore, $Pr[I_{r_i} = 0]$ is

$$Pr[I_{r_i} = 0] \leq Pr[\bigvee_{t=1}^{p_{r_i}^m} I_t^m = 0] \leq \sum_{t=1}^{p_{r_i}^m} Pr[I_t^m = 0] \tag{9}$$

$$\sum_{t=1}^{p_{r_i}^m} Pr[I_t^m = 0] \leq p_{r_i}^m \left(1 - \frac{\mu(B_{\rho_m/2}(q_i^m))}{\mu(C_{m_{free}})}\right)^m \tag{10}$$

The RHS of last inequality (Equation 8) can now be written as

$$\leq (p^b - k)(1-\sigma_1\rho_b^{d_b})^n + k(1-\sigma_1\rho_r^{d_b})^n + \sum_{i=1}^{k} p_{r_i}^m (1-\sigma_2\rho_m^{d_m})^m \tag{11}$$

where $\sigma_2 = \mu(B_2(\cdot))/2^{d_m}\mu(C_{m_{free}})$ and $B_2(\cdot)$ is the unit ball in $\mathbb{R}^{\mathrm{d_m}}$. Using the relation $(1-\beta)^n \leq e^{-\beta n}$ for $0 \leq \beta \leq 1$, the above inequality then finally can be written as

$$\leq c_1 e^{-\sigma_1 \rho_b^{d_b} n} + c_2 e^{-\sigma_1 \rho_r^{d_b} n} + c_3 e^{-\sigma_2 \rho_m^{d_m} m} \tag{12}$$

where $c_1 = (p^b - k)$, $c_2 = k$, and $c_3 = \sum_{i=1}^{k} p_{r_i}^m$. The expression above converges exponentially to 0 as $n \to \infty$ and $m \to \infty$, hence showing the completeness of the HAMP algorithm with respect to a class of paths, i.e., H-path. However, given any arbitrary path in open $C_{bm_{free}}$, we can always create an H-path that lies in open $C_{bm_{free}}$ as shown in Fig 6. Hence HAMP is probabilistically complete.

Algorithm 5: Π^{bm} = HAMP-U(q_s, q_g, q_H^m)

Input: The start $q_s = ((\mu_s, \Sigma_s), q_s^m) = (q_s^b, q_s^m)$ and goal $q_g = ((\mu_g, -), q_g^m) = (q_g^b, q_g^m)$ configurations, 3D Map and q_H^m
Output: Path Π^{bm} from q_s to q_g with low goal covariance Σ_g

1 Roadmap G_{belief}^b := ConstBeliefRoadmap(q_H^m)
2 Append G_{belief}^b with nodes $\{n_s := (\mu_s, q_H^m)$, $n_g := (\mu_g, q_H^m)\}$, edges $\{\{e_{s,j}\}, \{e_{i,g}\}\}$, and *one* descriptor matrices $\{\{S_{1:T_{s,j}}\}, \{S_{1:T_{i,g}}\}\}$
3 Augment node structure with best path $p := \emptyset$, covariance $\Sigma := \emptyset$, RPATHS$^{[n_{adj}]} = \emptyset$ such that $n_i := \{(\mu, \Sigma), q_H^m, p,$ RPATHS$^{[\cdot]}\}$
4 $Q \leftarrow n_s := \{(\mu_s, \Sigma_s), q_s^m, \emptyset, \emptyset\}$
5 **while** $Q \neq \emptyset$ **do**
6 $n :=$Pop(Q)
7 **if** $n[q^b] = n_g[q^b]$ **then**
8 Continue
9 **end**
10 **for** ***all*** n' *of adj[n]* ***and*** $n' \notin n[p]$ **do**
11 Compute one-step update $\Psi' = \Psi \star S_{1:T_{n,n'}}$ where $\Psi = \begin{bmatrix} I & n[\Sigma] \\ 0 & I \end{bmatrix}$
12 $\Sigma' \leftarrow \Psi'_{12}$
13 **if** $tr(\Sigma') < tr(n'[\Sigma])$ ***or*** $tr(n'[\Sigma]) = \phi$ **then**
14 $(q_{new}^m, \pi_n^m) :=$SearchManipPath(n, n')
15 **if** $\pi_n^m \neq \emptyset$ **then**
16 $n[$RPATHS$^{[n']}] := \pi_n^m$
17 $n' := \{(-, \Sigma'), q_{new}^m, n[p] \cup \{n'\}, -\}$
18 $Q \leftarrow Q \cup \{n'\}$
19 **end**
20 **end**
21 **end**
22 **end**
23 $\pi_{n_g}^m =$SearchManipPathAtBaseGoal$(\hat{q}_g^m, q_g^m)$
24 $\Pi^{bm} \leftarrow$ traceback using $n_g[p]$ and $n_i[$RPATHS$^{[n_{adj}]}]$ such that $n_i, n_{adj} \in n_g[p]$
25 **return** Π^{bm}

6 The HAMP-U algorithm

Now we extend our HAMP algorithm to account for localization uncertainty associated with the mobile base pose. With the motion and sensor uncertainty, the state of mobile base is not precisely known and is represented by Gaussian belief. We assume that the motion of manipulator is quite accurate (a reasonable assumption, give the joint encoders are quite precise). The objective of HAMP-U is to find a collision-free path with low belief covariance at the goal. Like HAMP, the HAMP-U algorithm is a two stage process described in Algorithm 5. For brevity, we omit the expand roadmap portion in our pseudo code.

In the first stage (Line 1 of Algorithm 5), we first create a belief roadmap (instead of the standard probabilistic roadmap in the base pose space for the basic HAMP) for the mobile base with manipulator remain-

Algorithm 6: G^b_{belief} =ConstBeliefRoadmap(q^m_H)

Input: q^m_H
Output: Belief roadmap G^b_{belief}
1 Sample mean poses $\{\mu_i\}$ from $C_{b_{free}}$ using a standard PRM sampling strategy to build roadmap node set $\{n_i\}$ such that $n_i = q_i = (\mu_i, q^m_H) = (q^b_i, q^m_H)$
2 Create edge set $\{e_{ij}\}$ between nodes (n_i, n_j) if CollisionFree(n_i, n_j)
3 Build *one* descriptor matrices $\{S_{1:T}\} \forall e_{ij} \in \{e_{ij}\}$
4 **return** $G^b_{belief} = \{\{n_i\}, \{e_{ij}\}, \{S_{1:T_{ij}}\}\}$

ing in a fixed home configuration. The belief roadmap formation is explained in Algorithm 6. A naive approach to build the belief roadmap is to sample beliefs directly from belief space (μ, Σ). However, the biggest challenge is to ensure that the nodes are reachable. Therefore, the planner first samples a set of mean poses $\{\mu_i\}$ from $C_{b_{free}}$ using the standard sampling step in PRM algorithm (Kavraki et al, 1996). In our case, a sample mean pose is collision-free if mobile manipulator with mobile base positioned at μ_i and manipulator in home configuration q^m_H is collision-free. We add an edge e_{ij} between pairs (μ_i, μ_j) if a sequence of controls exists to move the mobile manipulator without collisions along the straight line between poses. We then simulate a sequence of controls and measurements along each edge. To achieve this, a motion model of the mobile base and a sensor model of the sensor is needed. We omit the precise details since they are standard implementation of extended Kalman filter (EKF). Our notation follows that of Prentice and Roy (2009). The effect of these models are essentially given by matrices G_t, V_t, H_t, the Jacobians corresponding to the motion model (w.r.t state variable and w.r.t control variable) and the sensor model (w.r.t state variable); and matrices W_t, Q_t, the noise covariance matrices for motion and sensing, respectively. The matrices G_t, $R_t = V_t W_t V_t^T$ and $M_t = H_t^T Q_t^{-1} H_t$, for each step along the edge e_{ij} are computed. BRM uses maximum likelihood observations and a covariance factorization to combine multiple updates along an edge $e_{n,n'}$ into a single transfer function, represented by a descriptor matrix $S_{1:T_{n,n'}}$. The matrix encodes the uncertainty along the edge.

The second stage (Lines 2-24 of Algorithm 5) is a search mechanism that searches the belief roadmap using a variant of standard breadth first search algorithm and SearchManipPath() routine in an intertwined manner, similar to HAMP except that the metric used is uncertainty along the path rather than the length of path. The search process uses a queue function for the expansion of (μ, Σ) nodes in a first-in, first-out order. Line 10 prevents cycling problems, where an adjacent node n' is only considered if it is not already in the mobile base path $n[p]$. Note that, in Lines 13-20, we only expand nodes if search algorithm has found a new posterior covariance Σ' such that some measure of uncertainty (we use the trace) for it is less than that for existing posterior covariance $n'[\Sigma]$ and there is a path for the mobile manipulator along an edge $e_{n,n'}$. It is also assumed that a node n' replaces any current queue member n' when pushed onto the queue in line 18.

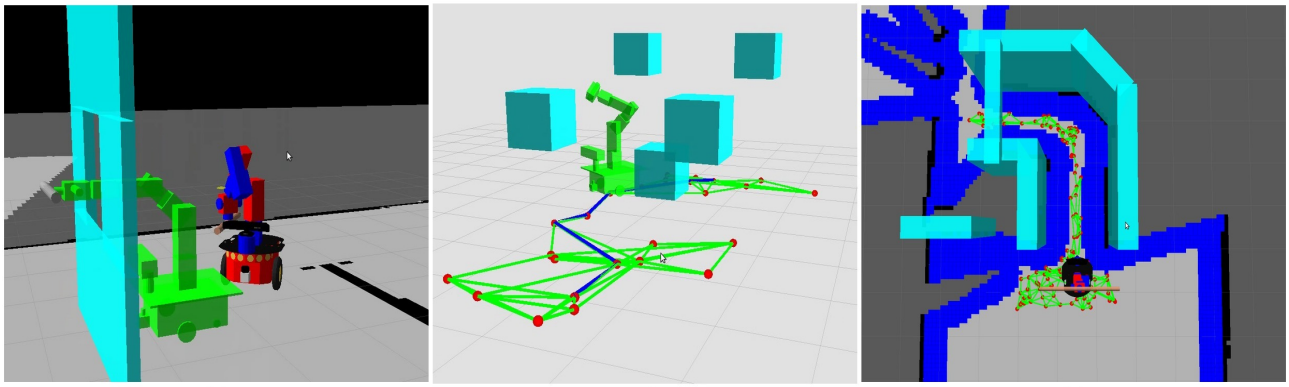
7 Results

We performed a series of simulations and real experiments on the SFU mobile manipulator roaming around in our lab to evaluate HAMP and HAMP-U. Our evaluation consisted of two main objectives: (a) to demonstrate the usefulness of the hierarchical search (HAMP) and its comparison with a full 9D PRM based on a set of performance criteria; (b) to show that by taking into account the uncertainty in the planning process, i.e., using HAMP-U, the paths generated are less likely to result in collision as compared to the basic HAMP.

The SFU mobile manipulator consists of a powerbot mobile base with a 6DOF Schunk powercube arm mounted on it. The world representation is computed offline by manually moving the robot around and using the two on-board sensors, an LMS100, a planar laser rangefinder that provides a 240° field-of-view at 30 Hz and an effective range of 18m; and a Kinect that provides 3D range data at 30 Hz and an effective range of 0.7-6m. In simulations also, we used the same mobile manipulator model with the corresponding sensors. As we constructed the maps (2D and 3D) by manually moving the robot, hence their is uncertainty in the mapping. We run our tests under linux (Ubuntu 10.10) on a Pentium dual core 2.5 Ghz computer with 4GB memory. We made use of portions of publicly available ROS (Quigley et al, 2009) and OMPL (Şucan et al, 2012) code as needed for implementation of our HAMP and HAMP-U algorithms.

7.1 World representation and collision checks

We use two types of map representation for collision check for efficiency reasons: a 2D costmap (inflates costs based on 2D occupancy grid and user specified inflation radius) and a 3D collision map (a set of occupied voxels) derived from an incrementally built global octree (Hornung et al, 2013). For efficiency reasons, collision detection for the whole mobile manipulator is accomplished in a two-stage process as follows. During initial construction of the roadmap (in routines ConstructBaseRoadmap() as well



(a) (b) (c)

Fig. 7 Simulation environments for scenarios C, D & E: (a) in this simulation the task is to pass a stick of 50 cm through a window of size 40cm×50cm, this figure shows the mobile manipulator with stick in start and goal configurations; (b) the mobile manipulator needs to navigate through 5 cuboid obstacles to reach the goal; (c) shows a narrow corridor (overhead view) of width 80cm with a tight round turn; the manipulator is required to move in the narrow corridor and negotiate the turn while carrying a 100 cm long stick.

Table 1 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 /#Armchks.	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

as ConstBeliefRoadmap()), the 2D projected footprint of the base is checked against the 2D costmap, and if it is collision-free then a 3D collision check is performed on the manipulator. We use height threshold to project 3D range data (from Kinect) to get a 2D costmap. During search (routines ComputeArmGoals() and SearchManipPath()), since the path is already collison-free with respect to the base and home configuration of the arm, only 3D collision checks are done for the arm. This strategy helps us to avoid unnecessary 3D collision checks (which can be expensive) without being overtly conservative.

7.2 Simulation results for HAMP

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 (Şucan et al, 2012) to implement full 9D PRM in an incremental way (single-query) along with random-bounce walk expansion strategy (Kavraki et al, 1996) to connect narrow passages. The key parameters we used in HAMP are as follows: $K_{Goals} = 3$, 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 Fig 1, while for scenarios C, D and E, the corresponding simulation environments are shown in Fig 7. 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.

Figures 8, 9 and 10 show snapshots of the simulation tests for scenarios B, C and D, respectively. In simulation tests corresponding to Fig 8 and Fig 10, the manipulator changes its configuration 3 times along the path and 1 time at the goal, while in Fig 9, the manipulator changes its configuration only at the goal.

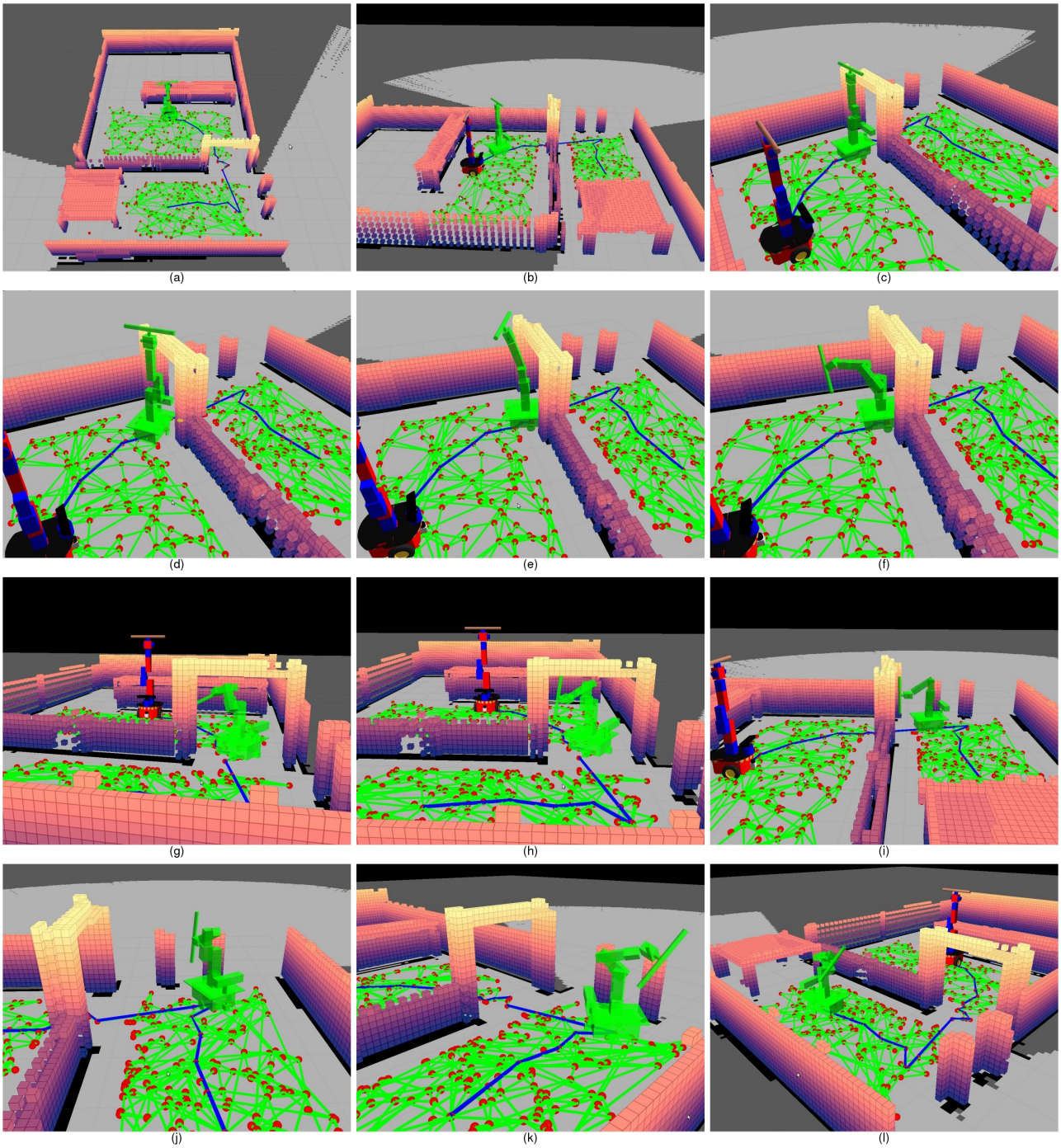


Fig. 8 HAMP simulation test for scenario B: This simulation shows the mobile manipulator carrying a 50 cm stick as payload and navigating from one side of the door to the other side. The corresponding base roadmap and base path (blue color) of H-path are shown as well. (a) mobile manipulator's start 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.

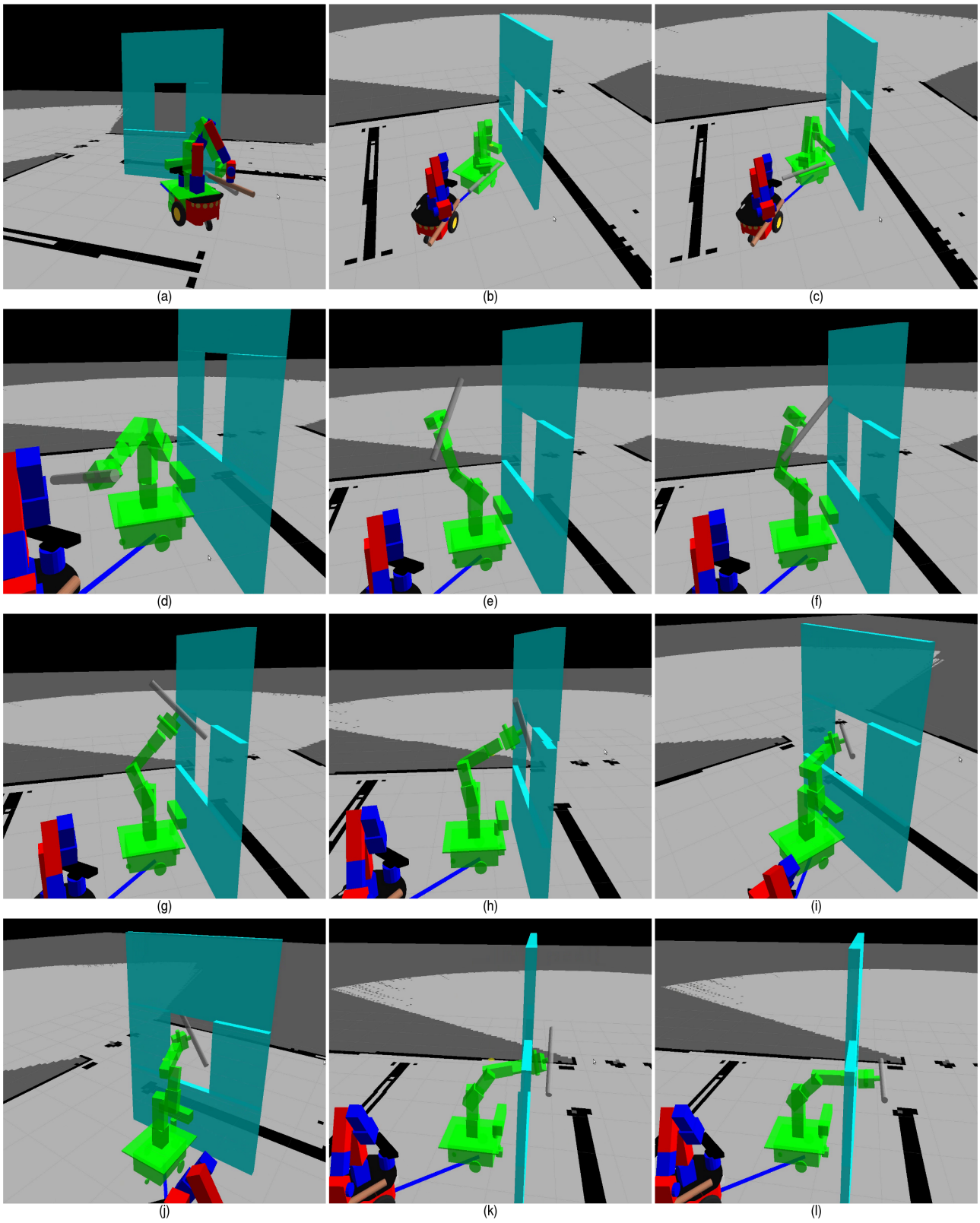


Fig. 9 HAMP simulation test for scenario C: This simulation shows the mobile manipulator passing a 50 cm stick through a window of 40cm×50cm (a) mobile manipulator's start configuration, here the arm is in compact state (folded) and the payload is held parallel along the side of the robot; (b), (c) mobile manipulator with the same arm configuration; (d), (e), (f), (g), (h), (i), (j), (k) show snapshots for arm reconfiguration step at goal, (l) shows the mobile manipulator in goal configuration.

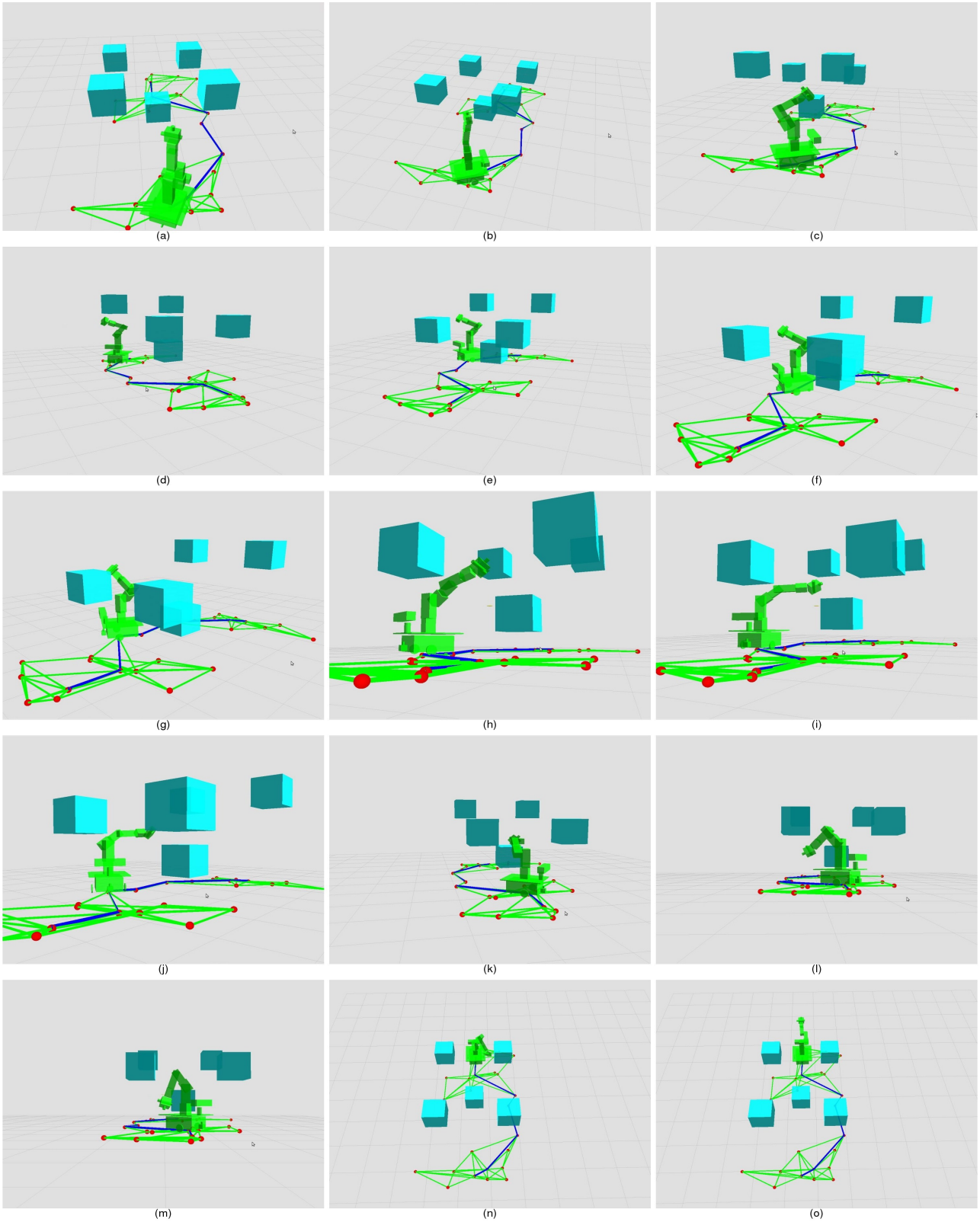


Fig. 10 HAMP simulation test for scenario D: This simulation shows the mobile manipulator navigating through 5 randomly placed cuboid obstacles (a) mobile manipulator's start configuration, here the arm is in vertically extended configuration; (b), (c) show the first reconfiguration step; (d), (e), (f), (g) mobile manipulator advances with the same arm configuration; (h), (i) show the second reconfiguration step; (j), (k) mobile manipulator advances with the same arm configuration; (l), (m) show the third reconfiguration step, (n) mobile base has reached the goal but not the arm, (o) shows the mobile manipulator's goal configuration, after the final reconfiguration step.

Videos corresponding to the simulation tests for scenarios A to E are available online at `http://www.sfu.ca/~vpilania/research.html` and also attached to this paper (Online Resources 1 to 4).

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 1. 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 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 ("#Armchks") along the edges in the base roadmap as illustrated in the column "#Reconfig/#Armchks" in Table 1. 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 Fig 11, 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) (Choset et al, 2005) . 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 re-configuration path (right figures). 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.

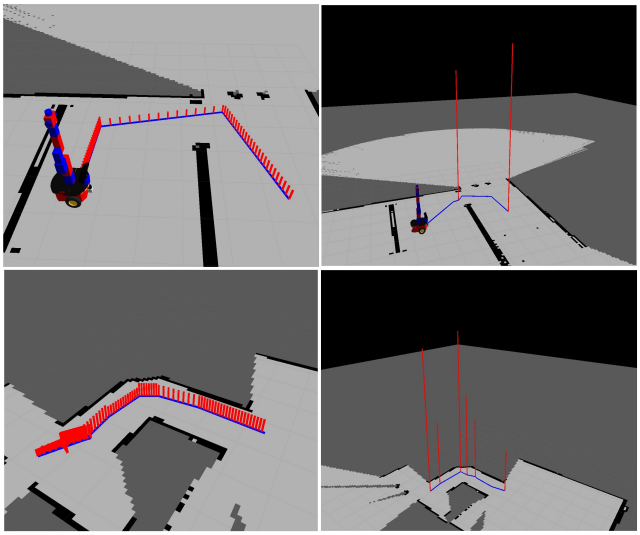


Fig. 11 This figure shows a comparison of manipulator motion (the height of red lines corresponds to the length of arm motion (in C_m)) along the base path (blue line) between full 9D PRM (left) and HAMP (right) for scenarios B (top) and E (bottom). Please see text for explanation.

7.3 Simulation results for tree versions of HAMP

We also evaluated the tree versions of HAMP with RRT and Bi-directional RRT (BiRRT) as the core subplanners for searching for both the base and the manipulator, respectively called HAMP-RRT and HAMP-BiRRT. In tree versions, it is not necessary to construct the entire tree first, hence stage 1 and 2 are essentially merged, i.e., for every new potential node to be added to the tree, SearchManipPath() is invoked from the nearest node. If it returns success then the new node is added to the tree along with corresponding updates at that node, else the node is rejected.

We compared HAMP-RRT with full 9D RRT and HAMP-BiRRT with full 9D BiRRT for scenarios A to E (as mentioned in Sec 7.2) and the corresponding results are presented in Tables 2 and 3. From the tables, we observe that for each of the scenarios A to E, full 9D RRT and full 9D BiRRT are outperformed

Table 2 Experimental results in 5 scenarios for mobile manipulator planning (HAMP-RRT vs. FULL 9D RRT).

	Permitted Time (s)	Planner	# Base Nodes		Total Arm Nodes		Time (s)		# Coll. Checks		#Reconfig /#Armchks.	B. Path Len. (m)	#Succ. /#Runs
			mean	s. d.	mean	s. d.	mean	s. d.	mean	s. d.			
A	40	HAMP-RRT	33	12	477	484	3.1	2.6	18k	15k	3/32	4.3	30/30
		RRT	1677	1379	N/A	N/A	7.7	6.9	45k	16k	N/A	5.9	30/30
B	40	HAMP-RRT	34	28	771	730	5.7	4.1	31k	21k	5/33	4.5	30/30
		RRT	2511	1208	N/A	N/A	13.8	7.3	68k	37k	N/A	6.5	30/30
C	120	HAMP-RRT	13	4	4338	2675	16.4	7.2	92k	43k	4/13	1.1	30/30
		RRT	5924	634	N/A	N/A	52.6	8.7	233k	56k	N/A	4.6	8/30
D	70	HAMP-RRT	27	13	989	778	9.0	4.3	53k	36k	12/26	3.2	30/30
		RRT	3218	756	N/A	N/A	18.2	8.1	96k	41k	N/A	4.2	30/30
E	300	HAMP-RRT	36	25	1379	1097	6.5	5.0	44k	33k	10/35	3.0	30/30
		RRT	3814	1097	N/A	N/A	23.4	9.3	127k	36k	N/A	4.7	30/30

Table 3 Experimental results in 5 scenarios for mobile manipulator planning (HAMP-BiRRT vs. FULL 9D BiRRT).

	Permitted Time (s)	Planner	# Base Nodes		Total Arm Nodes		Time (s)		# Coll. Checks		#Reconfig /#Armchks.	B. Path Len. (m)	#Succ. /#Runs
			mean	s. d.	mean	s. d.	mean	s. d.	mean	s. d.			
A	40	HAMP-BiRRT	21	11	309	336	2.4	1.2	15k	6k	2/20	4.4	30/30
		BiRRT	602	244	N/A	N/A	2.6	1.3	17k	7k	N/A	5.9	30/30
B	40	HAMP-BiRRT	23	19	663	528	4.5	2.7	24k	11k	4/23	4.5	30/30
		BiRRT	1447	591	N/A	N/A	8.1	3.4	42k	17k	N/A	6.6	30/30
C	120	HAMP-BiRRT	5	3	739	156	1.8	1.1	11k	5k	2/5	1.0	30/30
		BiRRT	346	185	N/A	N/A	2.2	1.1	13k	7k	N/A	4.6	30/30
D	70	HAMP-BiRRT	19	16	617	499	5.9	1.8	39k	12k	7/19	3.3	30/30
		BiRRT	1083	369	N/A	N/A	6.1	2.1	40k	22k	N/A	4.3	30/30
E	300	HAMP-BiRRT	30	23	1235	756	6.2	4.4	41k	27k	9/33	3.0	30/30
		BiRRT	1371	481	N/A	N/A	6.6	2.5	50k	38k	N/A	4.2	30/30

Table 4 Results for HAMP vs. HAMP-U (30 Runs).

	% times a given path results in coll.	
	HAMP	HAMP-U
A	16.6 %	3.3 %
B	46.6 %	10.0 %
C	63.3 %	50.0 %
D	56.6 %	16.6 %
E	73.3 %	43.3 %
F	33.3 %	6.6 %

by the corresponding tree versions of HAMP. Although the planning time difference is not that significant between HAMP-BiRRT and full 9D BiRRT. Similar to full 9D PRM, the computed path for the mobile manipulator using full 9D RRT or full 9D BiRRT also result into undesired and excessive motions for the manipulator even after applying a post processing smoothing filter. It is also important to note that as compared to HAMP, HAMP-RRT takes less time to plan a path and the planning time is further reduced in HAMP-BiRRT. This also holds true in case of full 9D PRM, full 9D RRT and full 9D BiRRT. In scenario C, full 9D RRT was only able to find a path in 8 trials out of 30. This is because there the goal was located in a very narrow region. In such cases BiRRT helps to find a path quickly as evident from the Table 3. Clearly, for the case with no uncertainty, HAMP-BiRRT is the planner of choice, because not only it is fast, it also avoids unnecessary motions of the arm.

However, for planning with uncertainty, tree versions of HAMP are not suitable, because there is no optimization possible with respect to uncertainty (note that RRT gives a unique path). One could get multiple paths via multiple runs of HAMP-RRT, however, this is likely to be computationally more expensive than the PRM version of HAMP (van den Berg et al, 2011). HAMP-BiRRT is, of course, not applicable because the uncertainty at the goal is not known in advance (it in fact depends on the path taken to the goal), hence one can not grow a tree from the goal, as needed in HAMP-BiRRT.

7.4 Simulation results for HAMP-U

We tested HAMP-U on 6 different scenarios in simulation and compared the outcomes with HAMP. Our main objective is to show that as a result of embedding BRM in our mobile manipulator planner (HAMP), the paths generated by HAMP-U are less likely to result in collision and are safer to execute as compared to HAMP. Scenarios A to E are same as explained in Sec 7.2, the only difference is instead of 50 cm stick we used 120 cm stick for scenario B. Fig 12 shows a sequence of snapshots for one of the simulation tests for scenario B, where the mobile manipulator needed to navigate with a stick of 120 cm. It shows that even with arm folded in most compact state (home configuration) at the start with the payload held parallel to the side of the base, the arm configuration still needs to be changed in order to reach the goal (as shown in Fig 12 (c), the mobile manipulator can not make a turn due to long payload collision with the wall or the gate). In scenario F, the mobile manipulator needed to navigate through 3 doorways of varying heights and widths to reach the goal. The corresponding simulation videos are available online at `http://www.sfu.ca/~vpilania/research.html`

A comparison between HAMP and HAMP-U for scenarios A to F is presented in Table 4. We generated

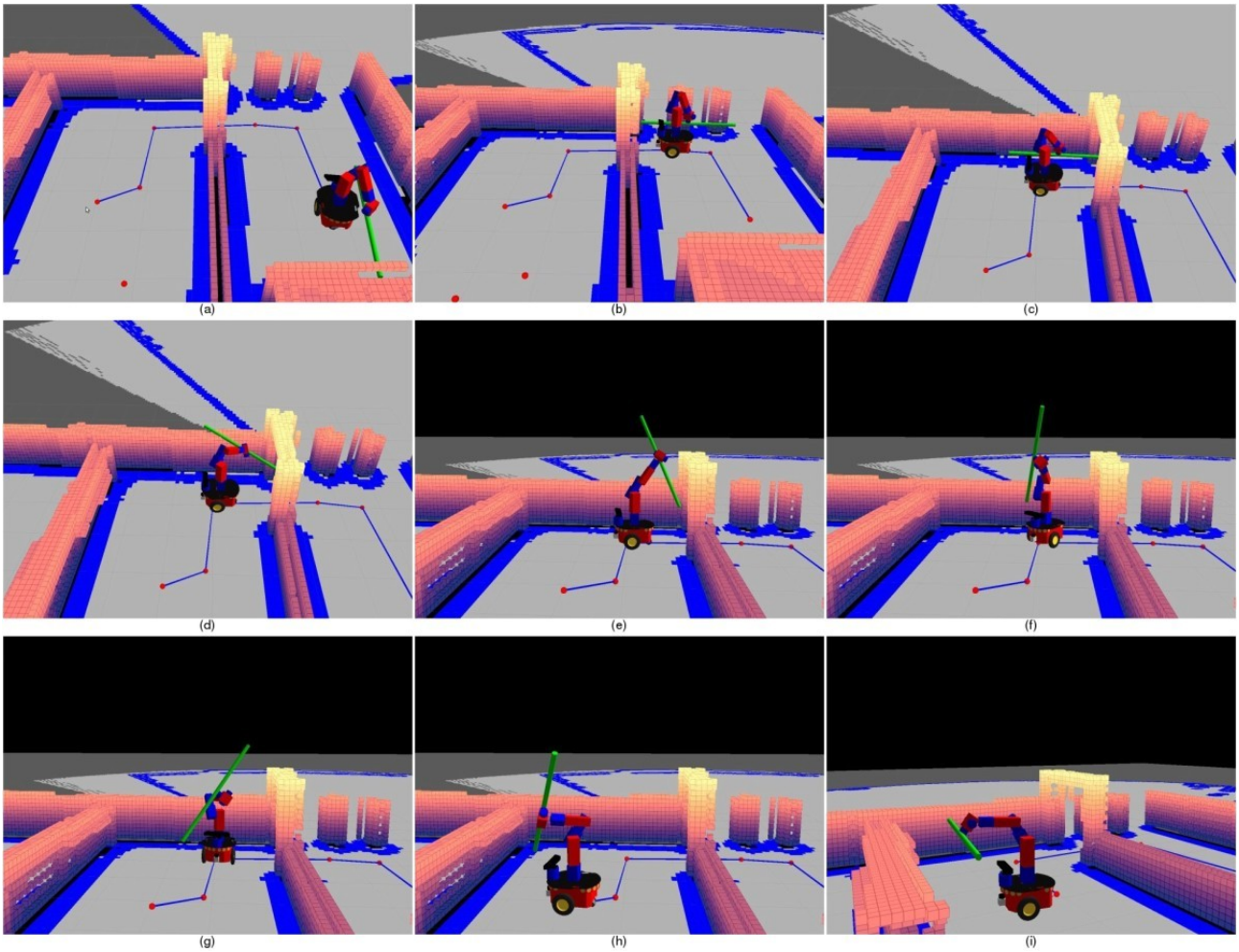


Fig. 12 HAMP-U simulation test for scenario B: The mobile manipulator carrying a 120 cm stick as payload. This simulation shows that even if you put the arm in compact state (folded), the arm configuration may still need to be changed in order to reach the goal (a) mobile manipulator's start configuration, here the arm is in compact state (folded) and the payload is held parallel along the side of the robot, (b) mobile manipulator with the same arm configuration, (c) in order to move along the path, arm configuration needs to be changed, as mobile manipulator can not make a turn due to long payload collision with wall or gate; (d), (e), (f), (g) show the reconfiguration step, (h) mobile base has reached the goal but not the arm, (i) shows the mobile manipulator's goal configuration, after the final reconfiguration step.

30 successful runs with HAMP and HAMP-U. Each path is then executed in simulation with artificially generated process and measurement noise, and the number of executions resulted in collision were counted. The percentage times a path generated by HAMP or HAMP-U results in collision with the obstacles upon execution is given in Table 4. We can see that the paths generated by HAMP-U are less likely to result in collision as compared to HAMP (which does not take into account the uncertainties). However, in HAMP-U for scenarios C and E, the collisions are still significant. This is mainly because the corresponding environments are more cluttered, hence, even a minor deviation of the base pose from the planned one leads to collisions. As mentioned in the future work (Sec 8), these collisions in HAMP-U can be mitigated by incorporating the effects of base uncertainty (for instance by explicitly using collision probabilities) on the planned manipulator motions.

7.5 Experiments on SFU mobile manipulator

Figures 13 and 14 show the 3D map of our lab formed by pre-sensing the environment with kinect and snapshots of one of the several experiments performed with the SFU mobile manipulator in our lab. The same example is shown in the experimental video attached to the paper (Online Resource 5) and also available online at `http://www.sfu.ca/~vpilania/research.html`. Our experimental set-up consists of two gates, 2nd gate is just 10 cm above the minimum height that can be at-

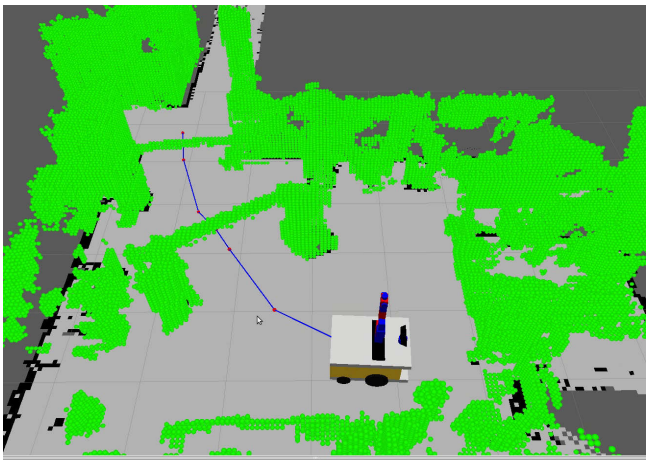


Fig. 13 The 3D world (collision map) is formed by presensing the environment with kinect and is shown in green above. The experimental set-up consists of two gates of different height.

tained by the manipulator. We used the extended arm configuration as initial configuration to illustrate the reconfiguration step. The planner runtime for the experiment, for which selected screenshots are shown in Fig 14 is 5 seconds. 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. In real experiments, for now, we are not able to demonstrate the pole example due to lack of gripper in SFU mobile manipulator.

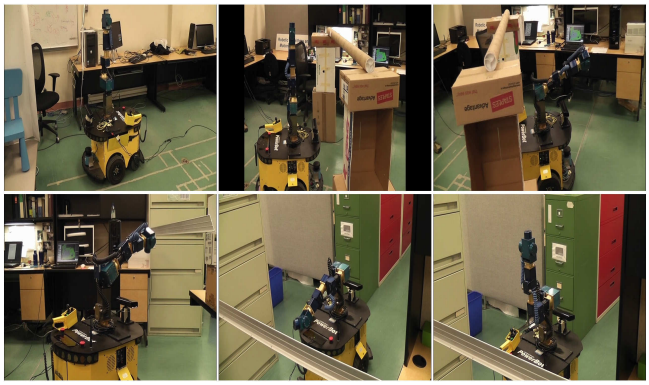


Fig. 14 Experiment on the real mobile manipulator. Photos are arranged from left to right in each row and then top to bottom. The mobile manipulator moves through two gates of varying heights and the manipulator reconfigures to a new configuration before crossing each gate.

8 Conclusion and future work

We presented a hierarchical and adaptive mobile manipulator planner (HAMP) which searches in two subspaces, the base sub-space and the manipulator subspace, 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 DOF 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. Additionally, we evaluated the tree versions of HAMP as well (i.e., HAMP-RRT and HAMP-BiRRT) by comparing them with full 9D RRT and full 9D BiRRT and observed the similar performance although the time saving for HAMP-BiRRT is modest as compared to full 9D BiRRT.

We then presented an extension to HAMP (HAMP-U) to account for localization uncertainty associated with the mobile base position. We showed that by incorporating base pose uncertainty within our overall HAMP algorithm, the paths generated by HAMP-U are less likely to result in collision and are safer to execute. In the current implementation, the low base uncertainty due to belief space roadmap indeed makes mobile manipulator motion plans safer than otherwise, however, the base uncertainty is not incorporated in the manipulator motion plans.

We would like to extend our work in two ways. The first is to extend it to an incremental version where rather than build an off-line map of the environment, as we do in the current implementation, the map is incrementally built as the mobile manipulator moves around, i.e., simultaneous planning and mapping (Yu and Gupta, 2004; Torabi et al, 2007), by incorporating an appropriate view planner. Second is to incorporate the effects of base uncertainty on the manipulator motion - for example, by taking into account the collision likelihood of such motions that is directly affected by the base uncertainty. This was done for fixed base position in (Huang and Gupta, 2009), and we plan to extend it to along an entire mobile manipulator path. 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 as well.

Acknowledgement

The work has been partly funded by an NSERC Discovery Grant of Kamal Gupta. We would like to thank the anonymous reviewer for insightful comments that helped us improve the paper.

A preliminary and earlier part of this work was presented at HUMANOIDS 2014 conference.

References

Agha-mohammadi, A., Chakravorty, S., Amato, N. (2014). FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, *33*(2), 268–304.

Berenson, D., Kuffner, J., Choset, H. (2008). An optimization approach to planning for mobile manipulation. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1187–1192).

van den Berg, J., Abbeel, P., Goldberg, K. (2011). LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, *30*(7), 895–913.

Bouilly, B., Simeon, T., Alami, R. (1995). A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1327–1332).

Burns, B., Brock, O. (2007). Sampling-based motion planning with sensing uncertainty. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3313–3318).

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

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

Şucan, I. A., Moll, M., Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, *19*(4), 72–82, URL `http://ompl.kavrakilab.org`.

Fraichard, T., Mermond, R. (1998). Path planning with uncertainty for car-like robots. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 27–32).

Gochev, K., Safonova, A., Likhachev, M. (2012). Planning with adaptive dimensionality for mobile manipulation. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2944–2951).

Gonzalez, J., Stentz, A. (2005). Planning with uncertainty in position: an optimal and efficient planner. In: *Proc.of the IEEE International Conference on Intelligent Robots and Systems (IROS)* (pp. 1327–1332), Edmenton, Canada.

Guibas, L., Hsu, D., Kurniawati, H., Rehman, E. (2008). Bounded uncertainty roadmaps for path planning. (In: *Workshop on Algorithmic Foundations of Robotics (WAFR)*).

Hornung, A., Bennewitz, M. (2012). Adaptive level-of-detail planning for efficient humanoid navigation. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 997–1002).

Hornung, A., Phillips, M., Jones, E. G., Bennewitz, M., Likhachev, M., Chitta, S. (2012). Navigation in three-dimensional cluttered environments for mobile manipulation. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 423–429).

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, *34*(3), 189–206.

Hsu, D., Latombe, J. C., Kurniawati, H. (2006). On the probabilistic foundations of probabilistic roadmap planning. *International Journal of Robotics Research (IJRR)*, *25*(7), 627–643.

Huang, Y., Gupta, K. (2008). RRT-SLAM for motion planning with motion and map uncertainty for robot exploration. In: *Proc.of the IEEE International Conference on Intelligent Robots and Systems (IROS)* (pp. 22–26).

Huang, Y., Gupta, K. (2009). Collision-probability constrained PRM for a manipulator with base pose uncertainty. In: *Proc.of the IEEE International Conference on Intelligent Robots and Systems (IROS)* (pp. 1426–1432).

Kaelbling, L., Littman, M., Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99–134.

Kavraki, L. E., Svestka, P., Latombe, J. C., Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*(4), 566–580.

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

Kurniawati, H., Du, Y., Hsu, D., Lee, W. S. (2009). Motion planning under uncertainty for robotic tasks with long time horizons. (In: *Proc. of the International Symposium on Robotics Research*).

Kurniawati, H., Bandyopadhyay, T., Patrikalakis, N. (2012). Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots*, *33*(3), 255–272.

Lambert, A., Gruyer, D. (2003). Safe path planning in an uncertain-configuration space. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4185–4190), Roma, Italy.

Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University.

Lazanas, A., Latombe, J. C. (1995). Motion planning with uncertainty: a landmark approach. *Artificial Intelligence*, *76*(1-2), 287–317.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B. P., Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 300–307).

Melchior, N. A., Simmons, R. (2007). Particle RRT for path planning with uncertainty. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1617–1624), Roma, Italy.

Missiuro, P., Roy, N. (2006). Adapting probabilistic roadmaps to handle uncertain maps. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1261–1267).

Monastero, A., Fiorini, P. (2009). Target pose computation for nonholonomic mobile manipulators. In: *Proc.of the IEEE International Conference on Advanced Robotics* (pp. 1–8).

Nakhaei, A., Lamiraux, F. (2008). A framework for planning motions in stochastic maps. In: *Proc.of the International Conference on Control, Automation, Robotics and Vision (ICARCV)* (pp. 1959–1964).

Pineau, J., Gordon, G., Thrunn, S. (2003). Point-based value iteration: An anytime algorithm for pomdps. In: *International Joint Conferences on Artificial Intelligence* (pp. 1025–1032).

Platt, R., Tedrake, R., Kaelbling, L., Lozano-Perez, T. (2010). Belief space planning assuming maximum likelihood observations. (In: *Proceedings of Robotics: Science and Systems*)Zaragoza, Spain.

Prentice, S., Roy, N. (2009). The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, *28*(11-12), 1448–1465.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A. Y. (2009). ROS: an open-source robot operating system. (In: *Proc. Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA)*).

Roy, N., Thrun, S. (1999). Coastal navigation with mobile robots. In: *Advances in Neural Processing Systems 12* (pp. 1043–1049).

Rusu, R. B., Blodow, N., Marton, Z. C., Beetz, M. (2009). 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)* (pp. 1–6).

Saxena, A., Driemeyer, J., Ng, A. Y. (2008). Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, *27*(2), 157–173.

Scholz, J., Chitta, S., Marthi, B., Likhachev, M. (2011). 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.

Smallwood, R., Sondik, E. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operation Research*, *21*(5), 1071–1088.

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

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

Tanner, H. G., Kyriakopoulos, K. J. (2000). Nonholonomic motion planning for mobile manipulators. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1233–1238).

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

Vannoy, J., Xiao, J. (2008). Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes. *IEEE Transctions on Robotics*, *24*(5), 1199–1212.

Yamamoto, Y., Yun, X. (1994). Coordinating locomotion and manipulation of a mobile manipulator. *IEEE Trans Autom Control*, *39*(6), 1326–1332.

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

Yang, Y., Brock, O. (2010). Elastic roadmaps - motion generation for autonomous mobile manipulation. *Autonomous Robot*, *28*(1), 113–130.

Yao, Z., Gupta, K. (2007). Path planning with general end-effector constraints. *Robotics and Autonomous Systems*, *55*(4), 316–327.

Yu, Y., Gupta, K. (2004). C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments. *International Journal of Robotics Research*, *23*(12), 1197–1223.