Mobile Manipulator Planning under Uncertainty in Unknown Environments

Vinay Pilania and Kamal Gupta

***Abstract*— We present a sampling-based mobile manipulator planner that considers the base pose uncertainty and the effects of this uncertainty on manipulator motions. The overall planner has three distinct and novel features: i) it uses the Hierarchical and Adaptive Mobile Manipulator Planner (HAMP) that plans for both the base and the arm in a judicious manner, ii) it uses localization aware sampling and connection strategies to consider only those nodes and edges which contribute toward better localization, iii) it incorporates base pose uncertainty along the edges (where arm remains static) and the effects of this uncertainty are considered on arm motion. We call this overall planner HAMP-BUA, where BUA stands for Base pose Uncertainty and its propagation to Arm motions. First we evaluate our planner in known static environments and show that it finds a safer path as compared to other variants where uncertainty is not considered at different levels as mentioned above. Next, we incorporate our planner within an integrated and fully autonomous system for mobile pick-and-place tasks in unknown static environments. A key aspect of our integrated system is that the planner works in tandem with base and arm exploration modules that explore the unknown environment. Our system is implemented both in simulation and on the actual SFU mobile manipulator and we present the corresponding results. It demonstrates a level of competency in exploring unknown environments for carrying out pick-and-place tasks that has not been demonstrated before.**

I. Introduction

A major challenge in motion planning is the uncertainty inherent in robot's control and sensing. Incorporating uncertainty into planning can improve the quality of computed plans which leads to much more reliable robot operation. Planning under uncertainty has made considerable progress over the past few years for mobile robots and unmanned aerial vehicles (UAVs) but still largely ignored for mobile manipulators. One key reason might be because most work usually take conservative approach, which is, to fold the arm to some "safe home configuration" and treat mobile manipulator essentially as a mobile robot where planners designed for them can be directly used to deal with uncertainty. However, this is a very limiting approach and fails in rather common scenarios where mobile manipulator can not move from start to goal without changing arm configuration as demonstrated in our previous work (Pilania and Gupta, 2014). Planners that consider uncertainty for mobile robots are not directly applicable to mobile manipulators in such scenarios since considering only the base pose uncertainty is not sufficient as it does not guarantee a safer path as demonstrated in our previous work (Pilania and Gupta, 2015a). Therefore, to get a safer path for the entire mobile manipulator, a more comprehensive approach should be taken, for example, in addition to base pose uncertainty, the effects of this uncertainty on manipulator motions must be considered as well.

Assuming that the motion of manipulator, in and of itself, is quite accurate (a reasonable assumption, given that joint encoders are quite precise), the uncertainty in mobile base position (which can be significant) has repercussions on the safety (i.e., collision status) of manipulator motions and grasping task (robotic hand). We group the effects of base pose uncertainty at two levels: Level 1 - the base pose uncertainty is considered only for mobile base motions and not translated to manipulator motion and grasping task; Level 2 - the base pose uncertainty is translated to manipulator motion. Our work considers uncertainty effects at both Level 1 and Level 2.

Previously, we designed a Hierarchical and Adaptive Mobile Manipulator Planner (HAMP) (Pilania and Gupta, 2014) that plans for both the base and the arm in a judicious manner - allowing the manipulator to change its configuration autonomously (arm reconfiguration) when needed if the current arm configuration is in collision with the environment as the mobile manipulator moves along the planned path. The underlying core-sub-planners were sampling based, one for the mobile base (in base pose space) and the other for the arm (in the arm configuration space), coupled in a loose manner (see Section III). This work did not consider uncertainty.

Our initial foray in incorporating uncertainty within HAMP framework was reported in Pilania and Gupta (2015a) and the resulting planner was called HAMP-U. It considered only Level 1 uncertainty in a limited way - it assumed that the robot is at the mean position (of belief, represented by uncertainty ellipse) and checked for collision from mean position to mean position between two belief nodes. So if the base deviates from the mean position during execution, collision may occur. Although HAMP-U helped in generating paths that were somewhat safer (less likely to collide) than those generated by HAMP, it was not adequately safe (a planned path for scenarios requiring frequent arm reconfiguration steps still has greater than 40 % chance of collision).

In this paper, we extend HAMP to comprehensively incorporate base pose uncertainty and its effects on the arm motion (i.e., both Level 1 and Level 2) thereby leading to safer paths for the entire mobile manipulator. We call the resulting planner HAMP-BUA where, BUA stands for Base pose Uncertainty and its propagation to Arm motions.

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

At the base level, it constructs a tree (in base pose space) using efficient localization aware sampling and connection strategies and propagates base pose uncertainty similar to Pilania and Gupta (2017) subjected to the collision probability threshold along the path and uncertainty threshold at goal. At Level 2, it searches for the reconfiguration paths along the planned base path segments by considering the effects of base pose uncertainty on the arm motions. The search for arm reconfiguration path in the arm C-space needs to explicitly incorporate the base uncertainty and the structure of HAMP lends itself to incorporating this uncertainty nicely via collision-probability constrained (CPC) planning approach for a fixed base manipulator with base pose uncertainty (Huang and Gupta, 2009). This CPC search is carried out at bases poses where the current arm configuration leads to collision if the base moves along a given edge.

It is important to note that incorporating Level 2 uncertainty requires embedding uncertainty related computations at different stages of path planning (placing samples, connecting them, etc.), which in turn requires costly operation of 3D collision checks, thereby greatly increasing the computation cost of the resulting planner. This is, we believe, one of the reasons, why planning under uncertainty for mobile manipulators largely ignores the manipulator and researchers simply choose to go with 2D planning for mobile base. Designing a reliable mobile manipulator planner is difficult to realise for real time applications unless one addresses the issue of extensive computations involved with mobile manipulator planning. And the real time aspect is particularly important, especially in unknown environments, which involves repeated and interleaved re-planning and exploration (with sensors) of the environment.

HAMP already avoids unnecessary 3D collision checks by first checking the 2D projected footprint of the base against the 2D costmap (obtained by projecting 3D range data from sensor up to a certain height), and only if it is collision-free, then a 3D collision check for the arm is performed. However, since incorporating Level 2 uncertainty significantly increases the number of collision checks needed, this is not enough to reduce the computation time and we need to find additional efficiencies.

One way is to look at next level, i.e., after doing collision checks for the sampled base point, do we really need to retain all the points (nodes) or the local paths (edges) connecting two points. A good decision at this level (before connecting the sampled point to the graph or tree) could help us improve the planner run time. In our previous work (Pilania and Gupta, 2015b, 2017), we demonstrated that localization aware sampling (LAS) and connection (LAC) strategies can be used to eliminate the nodes and the edges which do not contribute toward better localization. LAS puts more samples in regions where sensor data is able to achieve higher uncertainty reduction while maintaining an adequate number of samples in regions where uncertainty reduction is poor. While LAC first connects the new sample to a nearest node chosen based on an uncertainty metric and then the connections to other neighbouring nodes are rewired only if the new path to that node reduces the uncertainty. Our experimental results with a mobile robot (in 2D and no manipulator) showed that by using these smart strategies, the planning time can be reduced significantly with little compromise on the quality of path. Therefore, it was expected that the savings in planning time will be even higher for mobile manipulator, since the base cost of a 3D collision check is significantly higher (as we show in this paper) and smaller number of nodes and edges at the base level saves subsequent searches at the arm planning level. These strategies fall in the category of efficiency improvement at Level 1 and are incorporated within HAMP-BUA.

Overall, then HAMP-BUA incorporates following key components that result in safer paths that are efficiently searched for:

a) it propagates uncertainty from start to goal at base path search phase and uses collision probability for base path segment (edge) validation (Level 1 uncertainty)
b) it considers effects of base pose uncertainty while planning for arm motions (Level 2 uncertainty and efficiency)
c) a localization aware sampling strategy at sampling stage that results in fewer samples with little degradation in path quality (efficiency at Level 1), and
d) a localization aware connection strategy at connection stage that results in fewer edges without degrading performance (efficiency at Level 1)

No other existing mobile manipulator planner considers base pose uncertainty as comprehensively as HAMP-BUA does, and it results in significantly improved collision safety of the planned paths. Components **a)** and **b)** are algorithmic improvements on HAMP-U (Pilania and Gupta, 2015a) in incorporating uncertainty considerations at Level 1 and Level 2 and deal with safety. While HAMP-U did consider base uncertainty propagation, there were no collision probability constraint incorporated in the search process, even for the base. Similarly, the CPC-PRM was reported in an earlier conference version for a fixed base manipulator (Huang and Gupta, 2009). Our HAMP framework makes it possible to apply CPC-PRM notion over entire mobile manipulator motion plans from start to goal. Components **c)** and **d)** that deal with efficiency at Level 1 have been published before (Pilania and Gupta, 2015b, 2017), it was in the context of a mobile base in 2D; incorporating them within the mobile manipulator case is novel. LAS is modified for its use with planar range sensor (previously it was introduced for beacons) that involves additional ray tracing computations while LAC is augmented to suit the path search and tree expansion procedures of HAMP-BUA, i.e., new connections are made only if additional requirements of reconfiguration paths and collision probability thresholds are satisfied (see Section IV-C). Avoiding any one of the above components leads to degradation of either efficiency or robustness as we clearly show in our simulation results. For example,

if **a)** and **b)** are not considered, it leads to paths that are highly likely to result in collision when executed; if **c)** and **d)** are not considered then the mobile manipulator motion planning under uncertainty simply becomes much less efficient, especially for real applications.

We evaluate HAMP-BUA in known environment and show that it finds a safer path as compared to other variants where uncertainty is not considered at different levels, for example, not incorporating base uncertainty on manipulator plans, not respecting collision probability threshold along the edges. We also show that variants of this planner that do not use our localization aware sampling and connection strategies will take longer to find the same quality of path.

Furthermore, we incorporate HAMP-BUA within an integrated and fully autonomous system for end to end mobile [1] pick-and-place tasks in unknown static environments. The mobile manipulator is equipped with a Kinect sensor mounted on the base (provides an area scan depth map) and a eye-in-hand Hokuyo line scan sensor (mounted on the wrist of the arm with the last joint being used to obtain a area scan depth map) and uses these sensors to explore the environment. A key aspect of our integrated system is that the planner works in tandem with base and arm exploration (view planning) modules that explore the unknown environment. Note that unlike other implemented mobile manipulator planners, we assume unknown areas of environment as obstacles and not free and a region must be scanned free before the robot will move there. HAMP-BUA is central to this problem and can not be replaced by any other planning scheme. This is because each planning module of the system must consider uncertainty so that the paths are safe to execute. Therefore, a planner that does not consider uncertainty can not be used here. We chose this problem as it is considered a core sub-problem in service robotics and it is a first step toward putting theory into practice, and demonstrates the competence of HAMP-BUA planner.

We believe that our integrated and autonomous system scores many firsts: a) its competency is far superior to that of the existing integrated planning systems in that it explores the unknown environment while considering the unknown regions as obstacles, picks the object (once the object is deemed to be in the known region) and then further explores the environment with object in hand and places it at target location only after the place location is deemed to be in the known free region, b) it combines two different exploration schemes into one - uses frontier based exploration for the base and information gain maximization (in workspace) based exploration technique for the arm, c) it integrates scans from multiple sensors and then uses them for base and arm view planning, and finally d) in addition to above, as a working system, it integrates several modules - Octomap based environment representation, view planning for exploration, HAMP-BAU motion planner that accounts for base pose uncertainty and it effects on the arm motion, pick-and-place pipeline, localization, motion execution and online monitoring.

[1]The world "mobile" emphasizes that the mobile manipulator is required to move from one location to another.

II. Related work

In this section, first we review the related work on mobile manipulator planning under uncertainty, and then we consider the work concerning an integrated and autonomous systems that uses mobile manipulator for some tasks (for example, pick-and-place) in unknown environment.

A. Mobile Manipulator planning under uncertainty

Most of the previous work (Tan and Xi, 2001; Tanner and Kyriakopoulos, 2000; 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, there are two different approaches, one (Berenson et al., 2008; Hornung et al., 2012; Marder-Eppstein et al., 2010; Scholz et al., 2011) that folds the arm to some safe home configuration and plan for a 2D footprint of the mobile manipulator in a projected 2D environment representation. Hornung et al. (2012) uses a multi-layered 2D representation of both the robot and the environment but still the planning is carried out in 2D. While the second approach (Gochev et al., 2012; Pilania and Gupta, 2014; Vannoy and Xiao, 2008) is to reconfigure the arm if the current arm configuration is in collision with the environment as the mobile manipulator moves along the planned path. In second approach, there can be two possibilities, i.e., execute arm and base motions sequentially or continuously. In the latter, 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 as in Pilania and Gupta (2014). None of the above mentioned planning schemes deal with uncertainty.

The uncertainty typically originates from three sources: (i) motion uncertainty - uncertainty in a robot's motion often caused by factors such as wheel slippage, imperfect motion model, (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. Partially observable Markov decision process (POMDP) (Kaelbling et al., 1998) is a general mathematical 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 (Bai et al., 2014; Kurniawati et al., 2012, 2009; Pineau et al., 2003). A class of methods that carries robot state and associated uncertainty is an approximation to POMDP. Among them, a sub-class (Bouilly et al., 1995; Fraichard and Mermond, 1998; Lazanas and Latombe, 1995) assumes the presence of landmark regions in the environment where accumulated

motion uncertainty can be "reset". Another sub-class (Bry and Roy, 2011; Huang and Gupta, 2008; Lambert and Gruyer, 2003; Melchior and Simmons, 2007; Prentice and Roy, 2009; van den Berg et al., 2011) uses sampling-based methods (graph-based and tree-based) where uncertainty is propagated from start to goal. These methods are mainly demonstrated for mobile robots or UAVs and not directly applicable to mobile manipulators.

There are other works (Missiuro and Roy, 2006) that deal with mapping uncertainty. These approaches could be combined with our HAMP-BUA to incorporate the additional map uncertainty that would arise in the sensor fusion process due to base pose uncertainty.

B. Mobile manipulator based autonomous systems in unknown environment

Please note that individual exploration (view planning algorithms) techniques for either base or the arm are not the focus of this section. There is huge literature on that and a good review can be found in Torabi (2011). Here we review the related work on integrated and autonomous systems that use mobile manipulator for some application in unknown environment. Note that we consider unknown regions of the environment as obstacles and not collision-free regions (which can clearly be detrimental) as the assumption in Lehner et al. (2015). Furthermore, it does not use view planning to explore the environment. It just incorporates the sensor readings (from a 3D sensor mounted on the base and not acting as eye-in-hand) as the mobile manipulator moves along the path that follows end-effector trajectory. While it is true that in certain cases, if the environment is engineered to be free of obstacles, indeed one can make the assumption that unknown space is free. But in most environments (for example, consider a helper robot in an indoor environment with chairs and tables or hallways in buildings where there can be pillars in the middle of the open spaces), assuming unknown as free will lead to collisions, particularly for the arm - it may work just for the base because the planar lidars often mounted on the base will scan the unknown region in front before the base moves into the region, but it will not work for the manipulator since it can move into regions which have not been scanned by the sensor (Yu and Gupta, 2001). Dornhege and Kleiner (2011) searches for an object in the unknown environment using a planar range sensor mounted at end-effector but mobile base was fixed in their experiments. More recently, the winning entry to the Amazon Picking Challenge successfully performed several pick and place tasks and is described by Eppner et al. (2016). However, the task was quite constrained and several key choices were made to solve the specific task. The base motion was limited in that it was used simply to achieve a specific end effector pose, the environment was assumed known, and there was virtually no motion planning involved, with arm motions generated from pre-defined sequences of joint and task space controllers and on-line monitoring was used to guide task execution. The system proposed by Torabi and Gupta (2012a,b) is closer in spirit to ours in that it also integrates view planning and path planning for autonomously building a 3D model of an object in unknown environment. However it considers a completely decoupled approach for mobile manipulator planning and moreover, uncertainty is not considered at all. In fact, the lack of uncertainty consideration and the resulting plan failures in this system were a key motivation for us to incorporate uncertainty and hence the genesis of HAMP-BUA. There is lot of work related to environment exploration and mapping both in 2D and 3D either using mobile base (Stachniss et al., 2005) or UAVs (Shen et al., 2012). However, we have not come across any work where mobile manipulator is used to explore the unknown environment and achieve some tasks (for example, mobile pick-and-place).

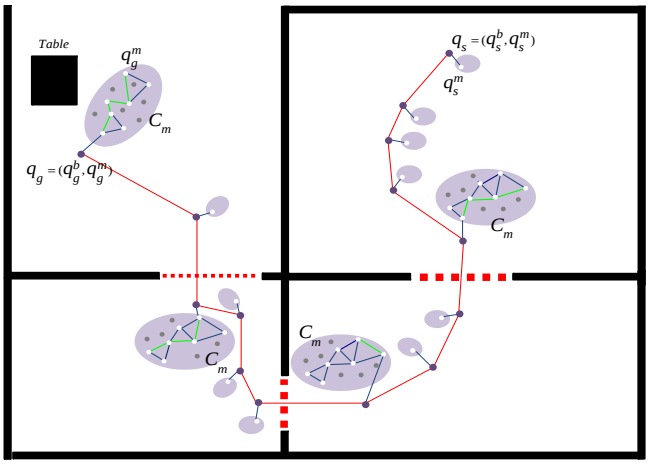

Fig. 1. A schematic illustrating the planned mobile manipulator path Π^{bm} given by HAMP algorithm. Please see text for explanation. This figure is taken from Pilania and Gupta (2014).

III. Background Information - HAMP Algorithm

Figure 1 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, 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.

Because of the hierarchical and adaptive approach, the nature of mobile manipulator path will have a specific structure as shown in Figure 1. We call this type of specific mobile manipulator path as an H-path. It consists of a set of mobile base poses (blue dots) and for each base pose, there is a corresponding manipulator reconfiguration path (white dots connected via light green edges).

IV. The HAMP-BUA Algorithm

A. Problem statement

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.

Given 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 as inputs, the objective (output) of our HAMP-BUA algorithm is to find a collision-free H-path Π^{bm} that respects collision probability threshold (CollProbTH) along the path and base pose uncertainty threshold (GoalUnctTH) at goal. Recall that H-path comprises sequential motions of base and arm, i.e., base moves with manipulator in static configuration, followed by a reconfiguration step where manipulator moves while base remains static.

B. General information

The algorithm operates on a set of nodes V and edges E, that define a tree in C_{bm}. Each node $v \in V$ has a base pose $v.q^b$, a manipulator configuration $v.q^m$, base pose estimate covariance $v.\Sigma$, a parent node $v.parent$, localization ability v.loc and reconfiguration path $v.\text{R}_{\text{PATHS}}^{[v_{adj}]}$ corresponding to child node v_{adj}. The base pose covariance prediction is implemented by a Propagate(e, Σ) routine that takes as arguments an edge and a covariance matrix at starting node for that edge, and returns a covariance matrix at the ending node. Routine LocalizationBiasedSample() outputs a random sample and its "localization ability". Localization ability of a sample is essentially a metric that reflects the sensor's ability to gather information were the robot to be at the sample point and is not the actual localization uncertainty at the sample. It is computed as the normalized difference of assumed covariance matrix at a sample point and the covariance matrix obtained after incorporating information gain (sensor readings). For conceptual ease, reader can view this as an efficient sampling scheme and details are in Pilania and Gupta (2015b). Earlier, LAS was introduced for beacon type of sensors. In this paper, we use the modified version which works with planar range sensor (for example, Hokuyo)

Algorithm 1: HAMP-BUA Algorithm

1 $v.q^b := q_s^b$; $v.q^m := q_s^m$; $v.\Sigma := \Sigma_s$; v.parent $:= \emptyset$; v.loc $:= tr(M)$; $v.\text{R}_{\text{PATHS}}^{[v_{adj}]} := \emptyset$
2 $V := \{v\}$; $E := \{\}$
3 **while** ! TimeUp **do**
4 **if** RNG$(0,1) <$ GoalBias **then**
5 $(q_{rand}^b, -) :=$ SampleBaseGoal(q_g^b)
6 **else**
7 $(q_{rand}^b, la) :=$ LocalizationBiasedSample()
8 $V_{near} := \{\text{Near}(V, q_{rand}^b)\}$; $\Sigma_{rand} := \emptyset$
9 **for all** $v_{near} \in V_{near}$ **do**
10 $e_{near} :=$ Connect$(v_{near}.q^b, q_{rand}^b)$
11 **if** CollisionFree2D(e_{near}) **then**
12 $\Sigma' :=$Propagate$(e_{near}, v_{near}.\Sigma)$
13 **if** $tr(\Sigma') < tr(\Sigma_{rand})$ ***or*** $tr(\Sigma_{rand})$=0 **then**
14 $v_{nearest} := v_{near}$; $e_{nearest} := e_{near}$
15 $\Sigma_{rand} := \Sigma'$
16 **if** $tr(\Sigma_{rand}) \neq 0$ **then**
17 $v' := \{q_{rand}^b, -, \Sigma_{rand}, v_{nearest}, la, -\}$
18 $(q_{new}^m, \pi^m) :=$SearchRPath$(v_{nearest}, v')$
19 **if** $\pi^m \neq \emptyset$ **then**
20 $cp :=$ ComputeCP$(v_{nearest}, e_{nearest}, q_{new}^m)$
21 **if** $cp <$ CollProbTH **then**
22 **if** *goal sample* **then**
23 **if** $tr(\Sigma_{rand}) <$ GoalUnctTH **then**
24 **if** RPathAtGoal(q_{new}^m, q_g^m) **then**
25 **return** *H-path* Π^{bm}
26 Continue (go to step 4)
27 Continue (go to step 4)
28 $v'.q^m := q_{new}^m$; $v_{nearest}.\text{R}_{\text{PATHS}}^{[v']} := \pi^m$
29 $V := V \cup v'$; $E := E \cup e_{nearest}$
30 **if** *node* v' *is added* **then**
31 **for all** $v_{near} \in V_{near} \setminus v_{nearest}$ **do**
32 $e_{near} :=$ Connect$(v'.q^b, v_{near}.q^b)$
33 **if** CollisionFree2D(e_{near}) **then**
34 $\Sigma' :=$Propagate$(e_{near}, v'.\Sigma)$
35 **if** $tr(\Sigma') < tr(v_{near}.\Sigma)$ **then**
36 $(q_{new}^m, \pi^m) :=$SearchRPath(v', v_{near})
37 **if** $\pi^m \neq \emptyset$ **then**
38 $cp =$ ComputeCP$(v', e_{near}, q_{new}^m)$
39 **if** $cp <$ CollProbTH **then**
40 $v_{parent} :=$Parent(v_{near})
41 v_{near}=$\{-, q_{new}^m, \Sigma', v', -, -\}$
42 $v'.\text{R}_{\text{PATHS}}^{[v_{near}]} := \pi^m$
43 E=$E \setminus (v_{parent}, v_{near}) \cup e_{near}$
44 UpdateTreeBranch(v_{near})

and the modification involves additional ray tracing computations. The trajectory between two states can be computed by routine CONNECT(). In our case, both simulation and real experiments assume holonomic robot to demonstrate our planners. However, if the system dynamics requires nominal trajectory and stabilizing controllers (which is beyond the scope of this paper) then it can be accommodated in this routine.

We also require the following routines: NEAR$(V, v.q^b)$ returns every node within some ball centered at $v.q^b$ of radius $\rho \propto (\log(n)/n)^{1/d^b}$ where n is the number of nodes and d^b is the dimension of C_b (See Karaman and Frazzoli (2011)). Note that NEAR() uses Euclidean metric in C_b. RNG() is a random number generator that generates a number distributed uniformly over an interval.

C. Algorithm description

The HAMP-BUA algorithm is described in Algorithm 1. The tree is initialized with a single node with base pose q_s^b, manipulator configuration q_s^m, covariance Σ_s and its localization ability $tr(M)$ (trace of matrix M) from lines 1-2. Note that the localization ability of a sample is stored at the node and this information will be used by our localization aware sampling strategy and not anywhere else in the algorithm. M is a hypothetical covariance matrix (we use an identity matrix).

At each iteration of the while loop, the tree is updated by adding a new sample. If the bias is not toward goal (line 6), a new base pose q_{rand}^b is sampled using our localization aware sampling strategy (line 7) and then connected to the nearest node. To connect the new sample to a nearest node and other neighbouring nodes (rewiring), we use our localization aware connection strategy (Pilania and Gupta, 2017). The nearest node is chosen based on uncertainty metric and not distant metric as described from lines 8-15. For each neighbouring node within a ball, the uncertainty is propagated from it to the new sample given that the corresponding path is collision-free. At this stage only 2D collision checks are performed, i.e. base footprint is checked with 2D representation of the environment. The neighbouring node which gives minimal uncertainty at the new sample is selected as nearest node.

The connection from nearest node to the new node v' is made only if the local base path ($e_{nearest}$) connecting nearest node and new node satisfies two conditions: a) there exists a reconfiguration path at nearest node such that the resultant manipulator configuration is collision-free as the mobile manipulator moves along the local base path, b) the collision probability along the local base path is below the given threshold. This is explained from lines 16-29 (excluding lines 22-27 which are for goal sample and explained later). Note that the dashes, for example in line 17, represent that there is no change in the corresponding variables. The reconfiguration path condition is checked in lines 18-19 where a routine SEARCHRPATH(), described in Algorithm 2, searches for a reconfiguration path that considers base pose uncertainty. This is where core of Level 2 uncertainty comes into play. While the second condition is checked in lines 20-21 where a routine COMPUTECP(), described in Algorithm 3, computes the collision probability of the mobile base motion along the local base path with manipulator in last configuration of the reconfiguration path. This is where core of Level 1 uncertainty comes into play. If both the conditions are satisfied then the information is updated at nearest and new nodes (line 28) and the new node v' and corresponding edge are added to the tree (line 29).

If the new node v' is successfully added then the HAMP-BUA algorithm rewires the connection to other neighbouring nodes, i.e. if the new path to a neighbouring node via new node gives less uncertainty then the new path is retained (edge connecting new node to a neighbouring node is added) while the edge connecting a neighbouring node to its parent node (in the old path) is deleted. The rewire connection from a new node to a neighbouring node is made only if following conditions are satisfied: a) new path is less uncertain, b) there exists a reconfiguration path, and c) the collision probability is below the threshold. The rewire connection procedure is explained from lines 30-44. The first condition is checked in lines 34-35 while the other two conditions

Algorithm 2: $(q_{new}^m, \pi_n^m) =$ SEARCHRPATH(n, n')

Input: nodes n, n' along an edge $e_{n,n'}$
Output: reconfiguration path π_n^m at node n with $n.q^m$ as start and q_{new}^m as goal such that q_{new}^m is collision-free along an edge $e_{n,n'}$

$n'' := (n'.q^b, n.q^m)$
if COLLISIONFREE3D$(e_{n,n''})$ **then**
  $q_{new}^m \leftarrow n.q^m$ and $\pi_n^m \leftarrow \{n.q^m\}$
else
  goal$^m :=$ COMPUTEARMGOALS$(n, n',$ K$_{\text{Goals}})$
  while ! ARMPLANNINGTIMEUP ***and*** ! TIMEUP **do**
    Search for a manipulator path from $n.q^m$ to one of the goal configurations in goal^m using a planner that considers base pose uncertainty.
    $\pi_n^m :=$LazyCPC-PRM$(n.q^b, n.\Sigma, n.q^m,$goal$^m[i])$
    $q_{new}^m \leftarrow$goal$^m[i]$
return (q_{new}^m, π_n^m)

Algorithm 3: $cp =$ COMPUTECP(n, e, q_{new}^m)

Input: node n, edge e, manipulator configuration q_{new}^m
Output: collision probability cp along edge e

Uniformly sample k particles from Gaussian distribution with mean $n.q^b$ and covariance $n.\Sigma$ such that weight (probability) of i^{th} particle is w_i
while $i < k$ **do**
  Compute collision status c_i by simulating the actions along e starting at i^{th} particle instead of $n.q^b$ with manipulator in configuration q_{new}^m. Note: collision free $(c_i = 0)$ or in collision $(c_i = 1)$.
  $cp := cp + (c_i \times w_i)$

are checked in lines 36-37 and lines 38-39, respectively. If all the conditions are satisfied then rewiring is done from lines 40-43. After rewiring a neighbouring node, the tree branch from that node onwards is updated using a routine UPDATETREEBRANCH(). This includes reconfiguration paths, collision probability checks and uncertainty propagation (line 44).

If the bias is toward goal (line 4), then HAMP-BUA tries to connect the base goal pose to the tree. The chance of this happening is 5% as we use GOALBIAS $= 0.05$. Same treatment is carried out to this goal sample up to line 21, i.e., collision checks are performed, reconfiguration path is searched and collision probability is checked. Thereafter, HAMP-BUA ensures that the uncertainty achieved at base goal pose is below the threshold (line 23). Note that because of reconfiguration steps along the path, the achieved manipulator configuration at base goal pose would be different from the desired one q_g^m. As a result of that routine RPATHATGOAL() is used to reconfigure the manipulator to q_g^m. If above conditions are satisfied then H-path is returned else the while loop continues till the permitted time to compute the path is over.

D. Reconfiguration path

The reconfiguration path search is implemented in SEARCHRPATH() which is described in Algorithm 2. This routine is similar to the one that we defined for HAMP in Pilania and Gupta (2014) except one modification, i.e., instead of using a traditional manipulator planner, we use the Lazy-CPC-PRM algorithm (explained below) that considers base pose uncertainty. In brief, if the current manipulator configuration is collision-free along the edge then reconfiguration step is not required (lines 1-4), else collision-free (along edge) manipulator configurations are searched using a routine COMPUTEARMGOALS(). This routine is different from HAMP in the sense that the COLLISIONFREE3D() is now collision probability constrained. Then Lazy-CPC-PRM is used to plan a path from current manipulator configuration to any of the configurations computed by COMPUTEARMGOALS().

In routine RPATHATGOAL(), we do not need to search a manipulator configuration to plan a path for as we already know that the manipulator needs to be reconfigured to q_g^m. Therefore, Lazy-CPC-PRM is used to plan a path from current manipulator configuration to q_g^m.

We give a brief overview of the method, for details please refer to Huang and Gupta (2009). Lazy-CPC-PRM, uses a set of particles to represent the uncertainty where each particle represents a base pose associated with a weight, which indicates the likelihood of this particle being the true base pose. A path for the manipulator is no longer simply either in-collision or collision-free, but is associated with a collision probability. Lazy-CPC-PRM constructs a probabilistic roadmap (PRM) for the manipulator, formulates the query phase as a search for a shortest path with the added constraint that the collision probability is lower than a user defined threshold. The resulting path search problem is called the collision probability constrained shortest path problem (CPC-SPP), and the resulting framework, lazy collision probability constrained PRM or Lazy-CPC-PRM. Recall that standard lazy algorithm (Bohlin and Kavraki, 2000) first searches for a shortest path over the roadmap without considering their collision status. Then, the path is verified, i.e., checked for collision. If it is collision-free, success is reported, otherwise, there must exist an edge along the path that is in collision. This edge is deleted from the roadmap and the shortest path algorithm is applied again over the modified roadmap to acquire an alternative path for verification. However, this standard lazy algorithm is not applicable to CPC-SPP. Since the edges are now associated with collision probabilities, if the collision probability of a path is higher than the threshold, it does not necessarily mean that there must exist an edge along the path that violates the collision probability constraint. Hence, one can not simply remove an edge from the graph and search for an alternative path as in the classic Lazy-PRM case. Lazy-CPC-PRM combines a k-shortest path algorithm (Hershberger et al., 2007) with a label setting algorithm (Dumitrescu and Boland, 2002) - standard for constrained shortest path search - to get the next shortest path, resulting in significant efficiency.

E. Collision tests

3D collision checks are performed in routines SEARCHRPATH(), RPATHATGOAL() and COMPUTEARMGOALS(). Our way of performing 3D collision checks is different from full-blown checks for mobile manipulator: first we check the 2D footprint of the mobile base with 2D representation of the world, if that is collision-free then only the manipulator model is checked with 3D representation of the world (not the mobile manipulator model). Both COLLISIONFREE2D() and COLLISIONFREE3D() use the collision probability threshold to decide on collision-free or in collision. The specifics of collision checks and world representation are described in later section.

F. Collision probability

Collision probability is an important metric that tells about the vulnerability (to collisions if robot deviates) of a path, which in turn helps to select a safer path. This is even more important for a mobile manipulator planner where manipulator needs to reconfigure (in case of HAMP, this reconfiguration is at a finite set of base locations). A slight base deviation can lead to collision (of the manipulator with the obstacles in the environment) even in less cluttered environment.

There are a few ways to compute the collision probability (along an edge with uncertainty ellipses at start and end vertices): (i) Huang and Gupta (2008) samples particles from both the ellipses and perform collision checks along local paths obtained from "all such pairwise paths". If n particles are sampled from each ellipse then there will be n^2 paths to check. Therefore, the complexity is of the order of $O(n^2)$, (ii) van den Berg et al. (2011) uses ellipse transformation approach and it was for the simple case of a disc robot, (iii) monte-carlo approach - where the edge is discretized and

for each point along the edge the corresponding uncertainty ellipse is obtained. For each ellipse, particles are sampled and then collision status is checked at each particle. That gives the collision probability of one ellipse. For an edge, all the collision probabilities along the discretized edge are multiplied (hence complexity of $O(kn)$ where k is the number of discretized points along the edge, and n the number of particles as above), (iv) Agha-mohammadi et al. (2014) uses action simulation approach, i.e., the same sequence of actions (used to travel along an edge) is simulated at each particle obtained from uncertainty ellipse at start vertex. The collision probability is then sum of weights corresponding to the particles that result in collision. In our case, we compute the sequence of actions as the rotation required to orient the robot along the edge, followed by the translation required to reach the other end of the edge. However, in general, there could be a non-straight line path between two nodes, in that case the corresponding sequence of actions should be considered. The complexity of this approach is of the order of $O(n)$.

Since first and third approaches are computationally more expensive, as indicated by their respective complexity as mentioned above, and especially for a mobile manipulator because of 3D collision checks this may result in quite long computation times. The second approach was applied to the simple case of a disc robot. Therefore, we use the fourth approach. The collision probability computation using this approach is described in Algorithm 3.

V. Simulation results for HAMP-BUA

In this section, we evaluate our planner in known environments and provide simulation results. Our implementation is in C++ under linux and runs on a Pentium dual core 2.5 Ghz computer with 4GB memory. From the simulations, we want to demonstrate two things: (i) our planner computes safer plans for mobile manipulator as compared to the deterministic case where no uncertainty is considered albeit at the cost of computational time, and (ii) consideration of our localization aware sampling and connection strategies helps to find the same quality of path in lesser time.

To prove the first objective, we compared HAMP-BUA with its variants where uncertainty is not considered at different levels. Note that these variants still make use of our localization aware sampling and connection strategies (i.e., components **c**, **d** as stated in Sec I), however, differ in the consideration of uncertainty along the edges and on manipulator motions. Below, we briefly describe these variants.

1) HAMP-BUA$_0$: This variant does not consider the collision probability along the edges (part of component **a**) and therefore, the corresponding threshold (CollProbTH) is not maintained, i.e., lines 21 and 39 of Algorithm 1 hold true. Also, the effects of base pose uncertainty on manipulator motions (reconfiguration paths) are not considered (component **b**, Level 2), i.e., instead of Lazy-CPC-PRM (line 8, Algorithm 2), a standard manipulator planner, as in HAMP, is used.

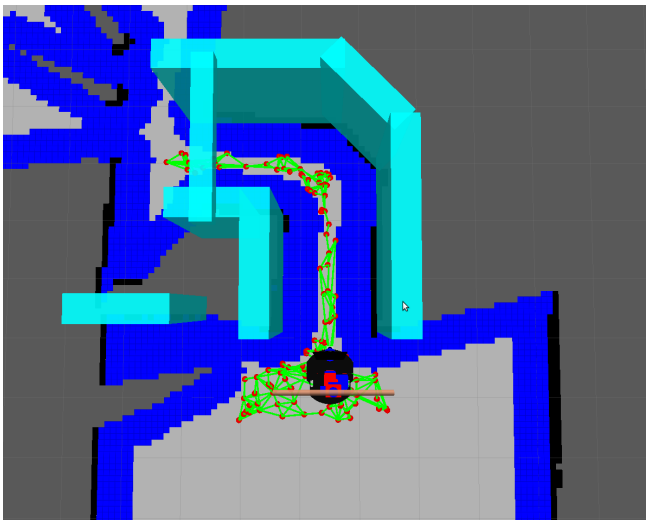

Fig. 2. Simulation environment for scenario C: shows a narrow corridor (overhead view) of width 80cm with a tight round turn, the manipulator is required to move in the narrow corridor and negotiate the turn while carrying a 100 cm long stick.

2) HAMP-BUA$_1$: This variant maintains the collision probability threshold for the base along the edges, however, the effect of base pose uncertainty on manipulator motion (i.e., Level 2 uncertainty) is not considered. Therefore, instead of Lazy-CPC-PRM (line 8, Algorithm 2), a standard manipulator planner, as in HAMP, is used.

A key reason to compare HAMP-BUA with its two variants, instead of just HAMP-BUA$_0$, is to show how the planner runtime increases as we incorporate the uncertainty at different levels and which component of the uncertainty consideration contributes what amount of increase in runtime and safety.

We evaluated HAMP-BUA and its variants in 3 different scenarios with varying level of complexity in simulation and compared the outcomes. In scenario A, the mobile manipulator was required to navigate from one side of the door to the other side while carrying a payload - a 50cm long stick. In B, the task required passing a 50cm long stick through a window of 40cm$\times$50cm. In scenario C (also shown in Figure 2), the mobile manipulator carrying a stick of 100cm, 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. Simulation environments corresponding to scenarios A, B are available in Pilania and Gupta (2015a).

For each scenario, we ran a planner 30 times to get 30 different paths with in permissible time limit of 180 seconds and measured for i) the average planner runtime and ii) the quality (in term of safety) of the computed path, i.e. if this path is executed then what is the probability that the mobile manipulator may collide with the obstacles. To compute the latter, we executed each path 20 times by artificially generating the noise (10% and 15%) in control commands and sensor measurements. Therefore, for a planner (in a

TABLE I

COMPARISON OF HAMP-BUA AND ITS VARIANTS (600 RUNS).

	HAMP-BUA$_0$		HAMP-BUA$_1$		HAMP-BUA	
	T. (sec)	%coll	T. (sec)	%coll	T. (sec)	%coll
A	8.2	15.0%	11.5	6.6%	31.1	0.8%
B	26.4	53.3%	28.0	45.0%	52.8	6.6%
C	58.5	48.0%	65.0	32.0%	168.0	5.2%

scenario), we monitored the number of times a collision has occured among 600 runs (30×20). These two properties are recorded for HAMP-BUA (and its variants) in Table I. "%coll" in the table denotes our second property. It tells the % of times a path results in collision.

From Table I, we observe that HAMP-BUA$_0$ takes less time to plan a path but at the cost of sacrificing path safety. Paths generated by HAMP-BUA$_0$ are highly prone to collisions. This is expected because at planning stage it does not respect the collision probability threshold along the edges and also the effects of base pose uncertainty are not considered on manipulator motions. As compared to scenario A, which is relatively a simple environment, the scenarios in B and C are complex and therefore, increasing collision risk. As we move toward HAMP-BUA$_1$, which does consider collision probability along the edges, the planning time increases but paths generated by HAMP-BUA$_1$ are less prone to collisions as compared to HAMP-BUA$_0$. Note that collision probability computation along the edges adds to planner runtime but then also gives safer plans. For scenario A, paths are 50% safer (as %coll decreased from 15.0 to 6.6) but that is not the case with scenarios B and C. For B, not much can be done with collision probability alone as the important task (of passing a long stick through the window) is achieved at base goal pose through reconfiguration path and HAMP-BUA$_1$ does not consider uncertainty on manipulator motions. While for C, %coll is reduced a lot while the rest can be achieved by consideration of uncertainty on reconfiguration paths. Finally, as we move toward HAMP-BUA, we can observe that the planner runtime is increased by 2 to 4 times while path safety is significantly improved. The increase in runtime by that amount is mainly due to Lazy-CPC-PRM, a manipulator planner that computes reconfiguration paths by considering base pose uncertainty. Separately, it is known that Lazy-CPC-PRM takes 2 to 20 seconds to compute a manipulator path depending on a threshold used there and the complexity of the environment. And for scenarios A, B and C, the average number of reconfiguration steps required were 5, 4, 20, respectively. That explains the increase in runtime.

Note that all the planners were successful in finding a path for each scenario (corresponding 30 trials) with in permissible time limit of 180 seconds, collision probability threshold of 0.08 and goal uncertainty threshold of 0.4. We also experimented by lowering these thresholds and observed that the rate of failure to find a path for HAMP-BUA increases with the decrease of these thresholds. For CollProbTH of 0.06 and GoalUnctTH of 0.32, the failure

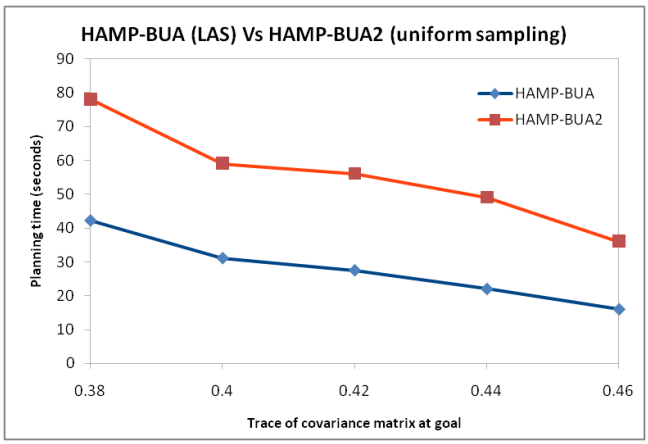

Fig. 3. HAMP-BUA Vs HAMP-BUA$_2$ by varying GOALUNCTTH while COLLPROBTH remains as 0.08 (for scenario A).

attempts (out of 30) were 1, 1, 3 for scenarios A, B, C, respectively. While for the thresholds of 0.06 and 0.28, the failure attempts increased to 1, 3, 6 for scenarios A, B, C, respectively.

To prove our second objective, we compared HAMP-BUA with its another variant where localization aware sampling strategy (component **c**) and a part of localization aware connection strategy (component **d**) were not used. In HAMP-BUA$_2$, a uniform sampling is used and the new sample is connected to the nearest node based on Euclidean distance and not on uncertainty based metric as in HAMP-BUA. Note that this variant still makes use of rewiring notion. For our simulations, we kept the collision probability threshold fixed (0.08) while varying the uncertainty threshold at goal. We ran our planner 30 times (for each GoalUnctTH) and averaged the planner runtime. The results are provided in Figure 3 for scenarios A. From the plot, it can be observed that HAMP-BUA takes less time to plan the same quality of path as a result of our smart strategies. On the other hand, the corresponding uniform sampling variant was also successful in finding a path that respects corresponding thresholds but take longer to reach there. It is also important to note that as we decrease the GoalUnctTH (looking for safer paths), the runtime difference between original planner and its variant increases. Therefore, our localization aware sampling and connection strategies help to reach toward well-localized path in shorter time by picking the right choice of samples and their connections (edges) which is highly useful and needed for mobile manipulator planning as it involves 3D collision checks.

A. World representation and collision checks

We use two types of map representation for collision check for efficiency reasons: a 2D world model (costmap) and a global 3D world model (collision map). The 2D world model is obtained by projecting the 3D map up to a certain height. At this stage, for evaluation of HAMP-BUA in known environment, it can be assumed that the 2D and 3D maps are known (provided). Later on, when we discuss

pick-and-place tasks in unknown environment, we explain in detail how these maps are constructed and the nitty-gritty involved. For efficiency reasons, collision detection for the whole mobile manipulator is accomplished in a two-stage process as follows. First, the 2D projected footprint of the base is checked against the 2D map, and if it is collision-free then a 3D collision check is performed on the manipulator only. This strategy helps us to avoid unnecessary 3D collision checks (which can be expensive) without being overtly conservative.

TABLE II
PARAMETERS USED IN HAMP-BUA.

Parameter name	Value
KGoals	3
ArmGoalsTimeUp	2 seconds
ArmPlanningTimeUp	6 seconds
GoalUnctTH	0.4
CollProbTH	0.08
DistTH	0.35 m
LocAbilityTH	83.3%
δ (Lazy-CPC-PRM)	0.2

B. Parameters and thresholds values

Table II shows the key parameters and their corresponding values that we used in HAMP-BUA and its variants. A few of these parameters are visible in the pseudo-code of the algorithms while rest of them are hidden inside few routines and previously developed approaches. For example: ArmGoalsTimeUp is a part of routine ComputeArmGoals(), precise detail of the pseudo-code can be found in Pilania and Gupta (2014). DistTH and LocAbilityTH are two thresholds used in routine LocalizationBiasedSample(), our localization aware sampling strategy. Effects of these two thresholds are well studied in Pilania and Gupta (2015b). Lazy-CPC-PRM (Huang and Gupta, 2009) also uses a threshold (δ) to compute a manipulator path for fixed uncertain base pose. Note that GoalUnctTH and CollProbTH are two different entities. Parameters CollProbTH and δ, both being collision probability, differ in their values. This is because the former is applied along the edges in the base roadmap (or tree) while the latter is applied to reconfiguration paths in manipulator C-space. These values are empirically chosen. One example of GoalUnctTH of 0.4 would be 10 cm (0.1 m) uncertainty each along x and y axes and 11 degree (0.2 radians) along base rotation (.1+.1+.2).

VI. Integrated and Autonomous System for Pick-and-Place Tasks in Unknown Environments

In this section, we report an integrated and autonomous system for mobile pick-and-place tasks in unknown static environments that uses our planner (in combination with view planning) to explore the environment and achieve a task. First, we provide a concise problem statement, followed by the description of our mobile manipulator system and the locations of different sensors and then the description of the system will follow.

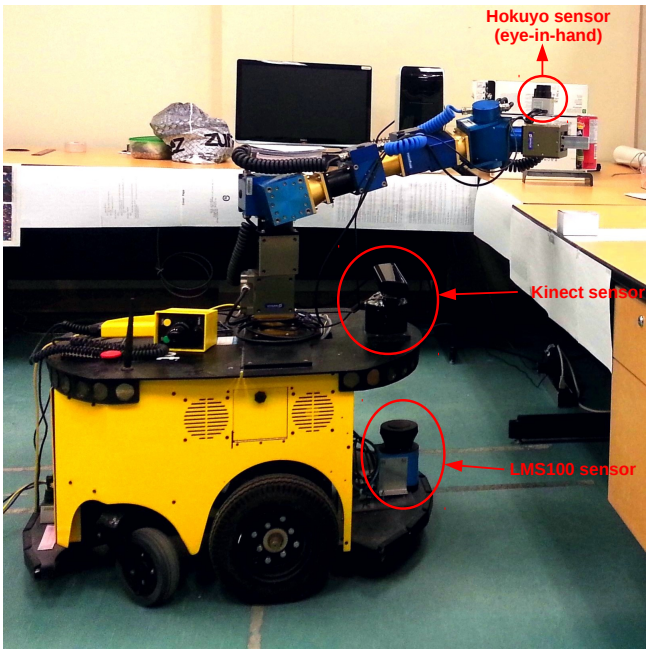

Fig. 4. The SFU mobile manipulator and mounting locations of different sensors.

A. Objective

Given global pick and place end-effector poses, the task of the autonomous mobile manipulator is to explore the environment, pick the object once it is in the known region and then explore the remaining environment (if needed) with object in hand, to place it at the target location (place pose). We assume that very small region around mobile manipulator is known in the beginning but other than that the entire environment is unknown. In future the grasp would be decided by the system too but for now we give the grasp pose.

B. System components

The SFU mobile manipulator model, both in simulation and physical environment, consists of a powerbot mobile base, 6 DOF schunk powercube arm mounted on the base, 2-finger schunk gripper, LMS100 2D range sensor placed in front of mobile base (used for SLAM), Kinect 3D depth and image sensor mounted on the base, and a light weight hokuyo 2D range sensor mounted on the gripper (to act as eye-in-hand) as shown in Figure 4.

1) Why we use 3 different sensors ?: An eye-in-hand sensor (Hokuyo in our case) is better able to explore the environment regions that are otherwise occluded for base mounted Kinect. Since the eye-in-hand sensor can provide only line scans, therefore, at NBV-A (next best view of arm), the sensor rotates to make an area scan by collecting all the line scans during rotation. The second sensor (Kinect) is added for online monitoring of path execution, however, once it is there it also acts as an additional sensor for exploring the environment. This is also required because Hokuyo does not work well with black surfaces and most of the surfaces

in real environment (RAMP Lab) are black. Kinect as eye-in-hand sensor (instead of Hokuyo in order to speed up the exploration) can not sense upto 1 meter distance (near clipping) and therefore, can not scan nearby regions. The third sensor, LMS 100 mounted at the front bottom of the base, is used to localize the mobile base.

C. System architecture description

Our integrated and autonomous system architecture is described in Figure 5. The only inputs to the system are global (w.r.t. world frame) pick and place end-effector poses. For each given pose, the corresponding possible base poses and manipulator configurations are computed. Note that at current stage the collision status of these base poses and manipulator configurations can not be verified as the environment is unknown. The method to compute a valid base pose and manipulator configuration corresponding to an end-effector pose is described in Torabi (2011) and is briefly as follows. First, a base pose is randomly sampled from a disk region (bounded by the arm reachability) and then a given end-effector pose (pick or place) is checked for reachability from the sampled base pose using inverse kinematics.

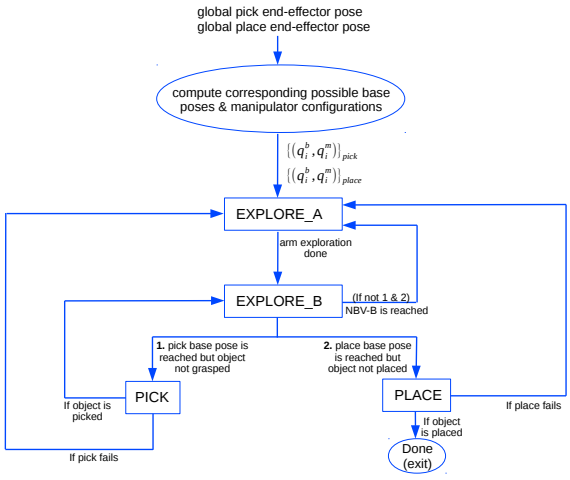

Fig. 5. An integrated and autonomous system for pick-and-place task in unknown environments.

At any instant of time the system can be in one of the four states: EXPLORE_A, EXPLORE_B, PICK and PLACE. In the beginning, the system starts in EXPLORE_A. This module uses eye-in-hand sensor to explore the local region around the mobile manipulator. Once the local region is fully explored, the system changes its state to EXPLORE_B. Broadly, one task of this module is to check for the reachability of pick and place base poses. If any of them is reachable then a path is planned to move the mobile manipulator to that base pose. Depending on success, the state is changed to PICK or PLACE. If none of these pick or place base poses is deemed reachable, then the EXPLORE_B module explores the environment by reaching to a NBV-B and state is then changed to EXPLORE_A. If at some point of time, the state is changed to PICK, i.e., pick base pose is reached, then the

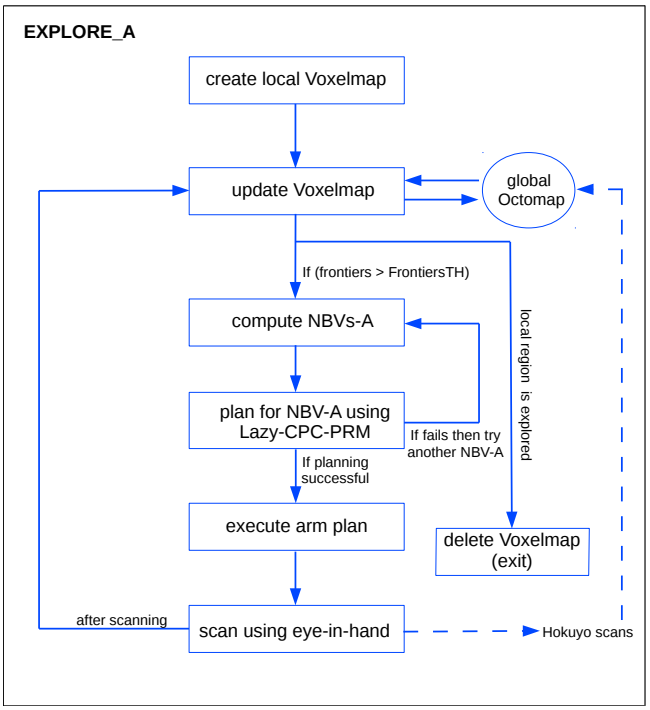

Fig. 6. EXPLORE_A module of the system.

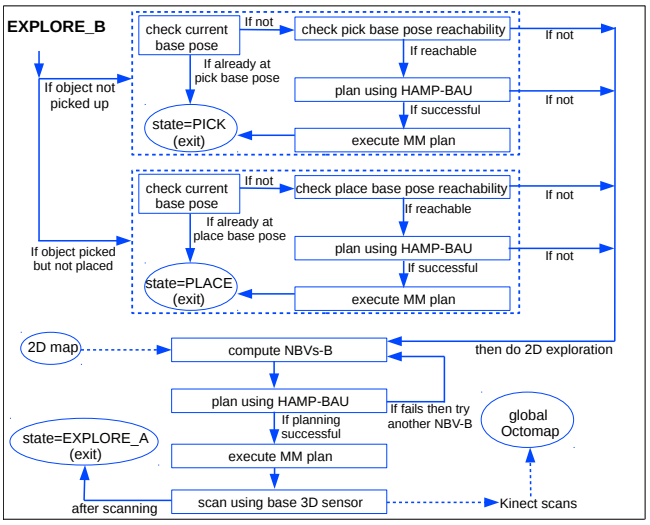

Fig. 7. EXPLORE_B module of the system.

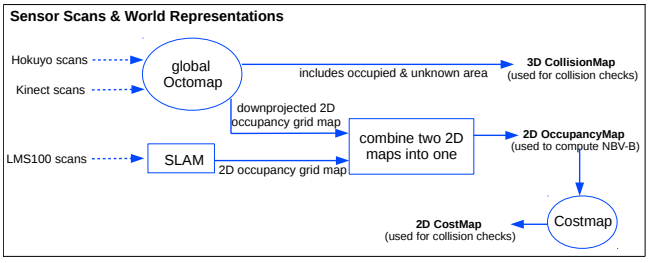

Fig. 8. Sensor scans and world representations (2D and 3D).

PICK module plans for pick end-effector pose and grasps the object. If grasping is successful then the state is switched back to EXPLORE_B to explore the remaining environment to complete the other part of objective, i.e., to place the object. As the system explores more and more environment (using EXPLORE_B and EXPLORE_A), the place base pose will be reachable at some point of time. Once the place base pose is reached then PLACE module is invoked to place the object at target location. The details of these modules are provided from Figures 6 to 7. We now describe these modules.

EXPLORE_A (see Figure 6) creates a local Voxelmap, a discretized grid representation of the 3D workspace as a 3D bitmap (with free as 0, obstacle as 1, and unknown as 2), at fixed base pose. Note that the boundaries of Voxelmap should be with in the reachability of end-effector. The status (occupied, free, unknown) of each voxel cell in the Voxelmap is updated by communicating with global Octomap (an octree representation of Voxelmap). Then the frontiers (free cells next to unknown) are computed. If the number of frontiers are below a given threshold then the arm exploration is aborted which states that the local region around mobile manipulator is fully explored. If not then arm view planning is invoked to compute a set of end-effector poses arranged in the priority order of information gain (which is the number of unknown voxels in the sensor FOV at the planned pose). We use MPV (Maximize Physical Volume) based algorithm for arm view planning as alluded to in Section II-B. The procedure to compute NBVs-A is as follow: arm view planning samples valid (IK exists and the solution is collision-free as well with in joint limits) end-effector poses and computes the information gain at each sensor pose by simulating the sensor model and then returns the set of these poses in the order of high information gain at the top. The end-effector is then moved to arm next best view (NBV-A) to take the scans using eye-in-hand Hokuyo sensor. Note that scans are inserted into the global Octomap and not the local Voxelmap. Lazy-CPC-PRM is used to plan a manipulator path for the IK solution of NBV-A as goal. After scanning, the local Voxelmap is updated and the procedure continues until the Voxelmap is explored or the maximum number of iterations are reached.

EXPLORE_B (see Figure 7) first checks, with in the known region of the environment, if it is possible to move the mobile manipulator to pick base pose or place base pose. If the object is not picked yet then the collision status of previously computed pick base poses is checked. If any of the base pose is found to be collision-free then HAMP-BUA is used to plan a path. If the pick base pose is reached (planning is successful) or the mobile base is already at pick base pose then the EXPLORE_B module simply switches the state to PICK. Similarly, if the object is grasped but not placed, the reachability of a collision-free place base pose is checked and if successful, the EXPLORE_B module aborts by switching the state to PLACE. However, if the planning fails or pick and place base poses are not collision-free with in the explored region then a base next best view (NBV-B) is reached to explore the environment using sensor (Kinect) mounted on the base. Again, the sensor scans are inserted into the global Octomap. After taking scans, the state is switched to EXPLORE_A to explore the local region, this time from a different base pose. To compute NBV-B, we invoke base view planning where we use frontier-based exploration (Yamauchi, 1997). The base view planning uses a 2D occupancy grid map (a discretized grid representation of the 2D workspace as a bitmap with free as 0, obstacle as 1, and unknown as 2) to compute NBV-B and this map comes from a series of steps as shown in Figure 8.

We now explain PICK and PLACE modules which are pretty standard in grasping domain. PICK module is invoked once the mobile manipulator has reached a base pose from where object can be grasped. Previously computed information (base poses and manipulator configurations) corresponding to pick end-effector pose (grasp pose) may not be useful any longer because due to the uncertainty in mobile base position, the mobile manipulator (basically mobile base) does not exactly reach the intended pick base pose. Therefore, either a new manipulator configuration needs to be searched from the reached base pose or a new valid grasp pose (and thereby the corresponding reachable manipulator configuration), with in the close proximity of already given grasp pose, needs to be computed. In case of latter (grasp adjustment), we use a simple approach as follows: Recall that a grasp pose is denoted as $p = [x, y, z, \alpha, \beta, \gamma]$, pose of end-effector frame (located in the center of gripper jaws) with respect to a fixed frame, for example, base frame of the manipulator. In simulation and real experiment examples, the new valid grasp pose is computed by varying γ while keeping other 5 parameters same. As mentioned earlier, this is a very quick and an ad hoc attempt to show the integrated system. For a valid grasp pose (collision-free IK solution exists), we also test if its pre-grasp and post-grasp poses are valid ones as well, which are typically 10 cm offset (back and up, respectively) from the intended grasp pose, see Leeper et al. (2010) for detail. Moreover, the end-effector motions from pre-grasp to grasp and grasp to post-grasp should be feasible. Once a valid grasp is found, a path is computed using Lazy-CPC-PRM to reach to the IK solution of pre-grasp pose and then followed by a straight line motion of the end-effector from pre-grasp to grasp pose. For more precise grasping, one can use reactive behaviour while moving the gripper from pre-grasp to grasp pose or a force-regulating controller that helps to close the gripper which takes feedback from tactile sensor in order to safely grasp an object (Ciocarlie et al., 2010; Leeper et al., 2010). However, in our implementation, we skip this due to the lack of tactile sensor in our gripper. PLACE module also follows the same steps but in reverse order. In future, our PICK module will be replaced by a systematic approach. For example, for cases where the object is known and can be visually tracked (by putting some markers), visual servoing with end-effector cameras can ensure the robust execution of grasps by incrementally correcting the position of the object relative to the gripper. An excellent survey on this subject can

be found in Kragic and Christensen (2003). For cases where the object and gripper can not be visually tracked, methods such as Ciocarlie et al. (2010), Chitta et al. (2012), Collet et al. (2011) can be incorporated. While in the presence of object pose uncertainty and high clutter, push-grasping approach in Dogar and Srinivasa. (2010) can be helpful.

D. HAMP-BUA for NBV-B

Note that HAMP-BUA is designed to plan a path from start to goal (base poses and manipulator configurations). However, to plan for NBV-B, we just have the goal base pose but the goal manipulator configuration is not known (neither required). Still, with minor modification, the planner can be used to plan a path for the mobile manipulator with NBV-B as a goal. For the modification, we skip line 24 in Algorithm 1, i.e., the reconfiguration step at goal base pose (NBV-B) is not needed.

E. Sensor scans and world representations

Figure 8 describes how the scans from different sensors are inserted and two (2D and 3D) world representations are formed. Hokuyo and Kinect scans are inserted into global Octomap (Hornung et al., 2013), a 3D world representation that maintains occupied, free and unknown regions. From the Octomap, we get a 3D collision map (a set of occupied and unknown voxels, all of the same size) and a down projected (up to a certain height) 2D occupancy grid map. This occupancy map is further fused with another 2D map obtained from SLAM module (which uses LMS100 scans for localization with map resolution of 0.05 m) to get a single 2D map. The fused 2D map, which shares information from all the sensors, is then used by base view planning to compute NBVs-B. We further input this map to the costmap module (assigns costs based on 2D occupancy grid and a user specified inflation radius) to get a 2D costmap. Costmap is an extension of occupancy map where a cost is assigned to each grid cell. The cost can depend on various factors, but in our context, it is in inverse proportion to the distance from obstacles. Inflation radius is a safety distance margin around the obstacles. The 2D costmap and 3D collision map are then used to perform collision checks as mentioned in Section V-A.

VII. Results

First, we separately evaluate EXPLORE_A module which is a key component of the system. If this module does not do its job properly then the system may not be able to complete the pick-and-place task. Thereafter, we provide full-fledged simulation and real experiment results for mobile pick-and-place task in unknown environment.

The task was to explore the unknown region within the boundaries of a local Voxelmap. We carried out 40 trials and for each trial we used different scenarios ranging from simple surrounding environment (no obstacles in the Voxelmap region) to cluttered ones like table with objects on it along with walls on other two sides. We set 10 seconds as maximum permitted time for Lazy-CPC-PRM to plan a path and monitored (at each iteration) the number of frontiers, voxelmap update time, time to compute NBVs-A and planner runtime. To give a clear picture of how much time is taken by each component of EXPLORE_A as the number of iterations approach to the maximum limit, we provide one trial result for the cluttered scenario in Table III. From the table, we can observe that the number of frontiers decreases below threshold with the increase of iterations. It is also important to note that the time to compute NBV-A increases drastically as the number of frontiers dropped to small numbers. This is because to find NBV-A for a small unknown region requires large number of samples which in turn requires many simulations of sensor model (ray-tracing) which is a time consuming step. In our 40 trials, EXPLORE_A succeeded in exploring the entire Voxelmap region all the time with in 15 iterations. On an average it took 11 to 15 iterations to explore the local region.

TABLE III

EXPLORE_A module test in cluttered environment.

	# frontiers	Voxelmap update T. (sec)	NBV-A T. (sec)	Planning T. (sec)
1	1498	0.0421	3.15	6.76
2	1649	0.0016	4.93	3.63
3	1562	0.6911	3.97	3.50
4	1395	0.2380	4.90	4.21
5	1380	0.5071	4.34	3.27
6	1226	0.0448	3.97	7.34
7	1191	0.0714	4.79	2.66
8	894	0.7044	4.58	2.20
9	728	0.0098	4.28	6.49
10	491	0.0756	8.74	2.39
11	151	0.4243	14.5	1.28
12	52	0.0427	29.6	1.35
13	24	0.0938	66.1	3.18
14	4	0.1619	-	-

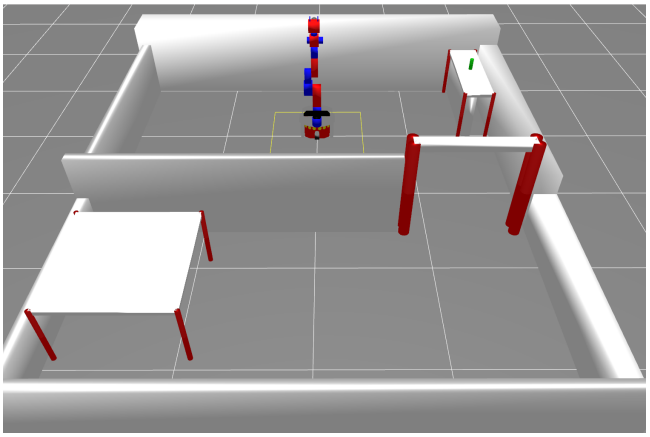

Fig. 9. Simulation environment for mobile pick-and-place task. The task is to explore the environment, pick the object (green colour) and place it on table located on the other side of the door.

A. Simulations

We now present simulation results for mobile pick-and-place task in unknown environment to demonstrate HAMP-BUA. The simulation environment is shown in Figure 9. We ran our integrated and autonomous system in this en-

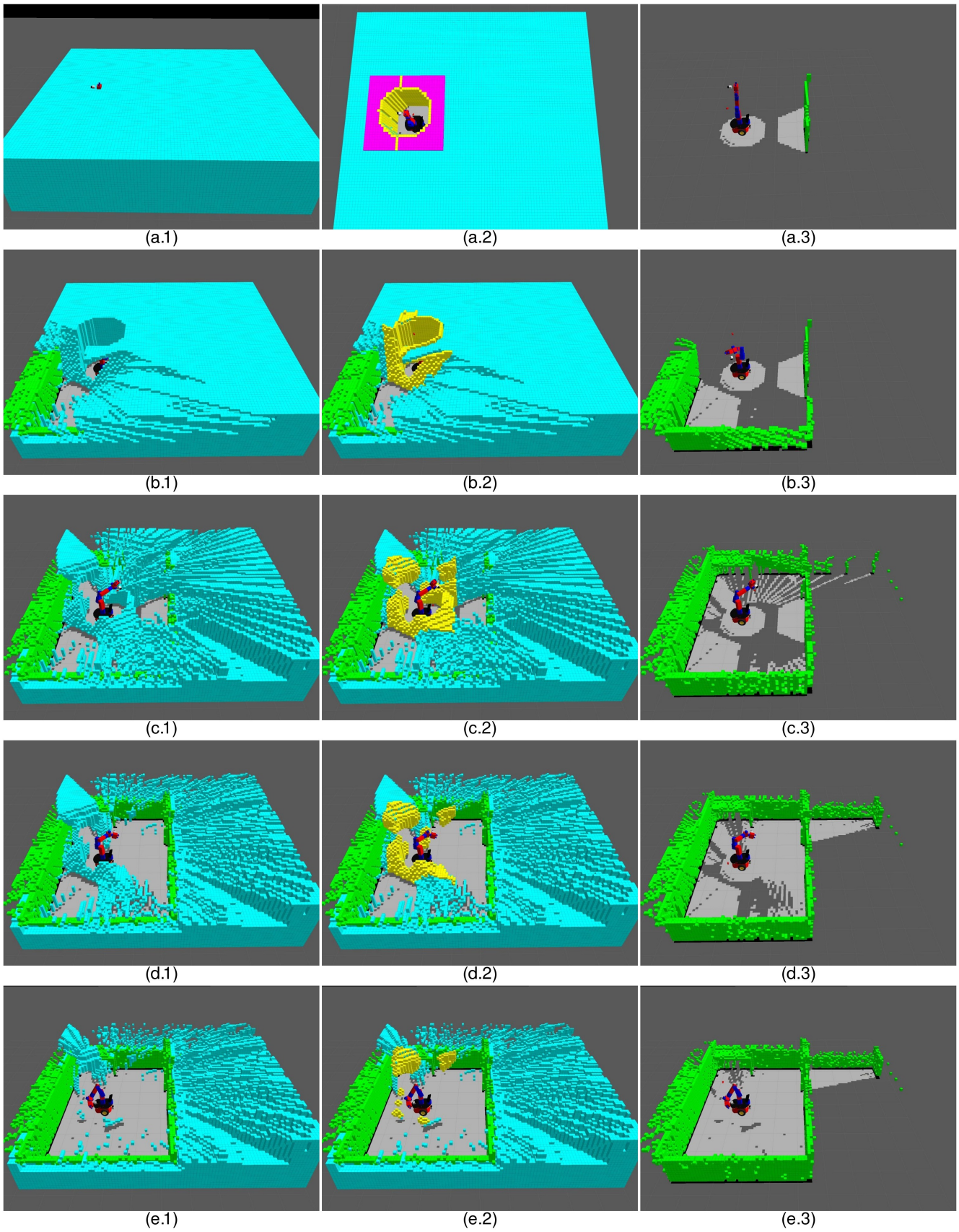

Fig. 10. Continued...

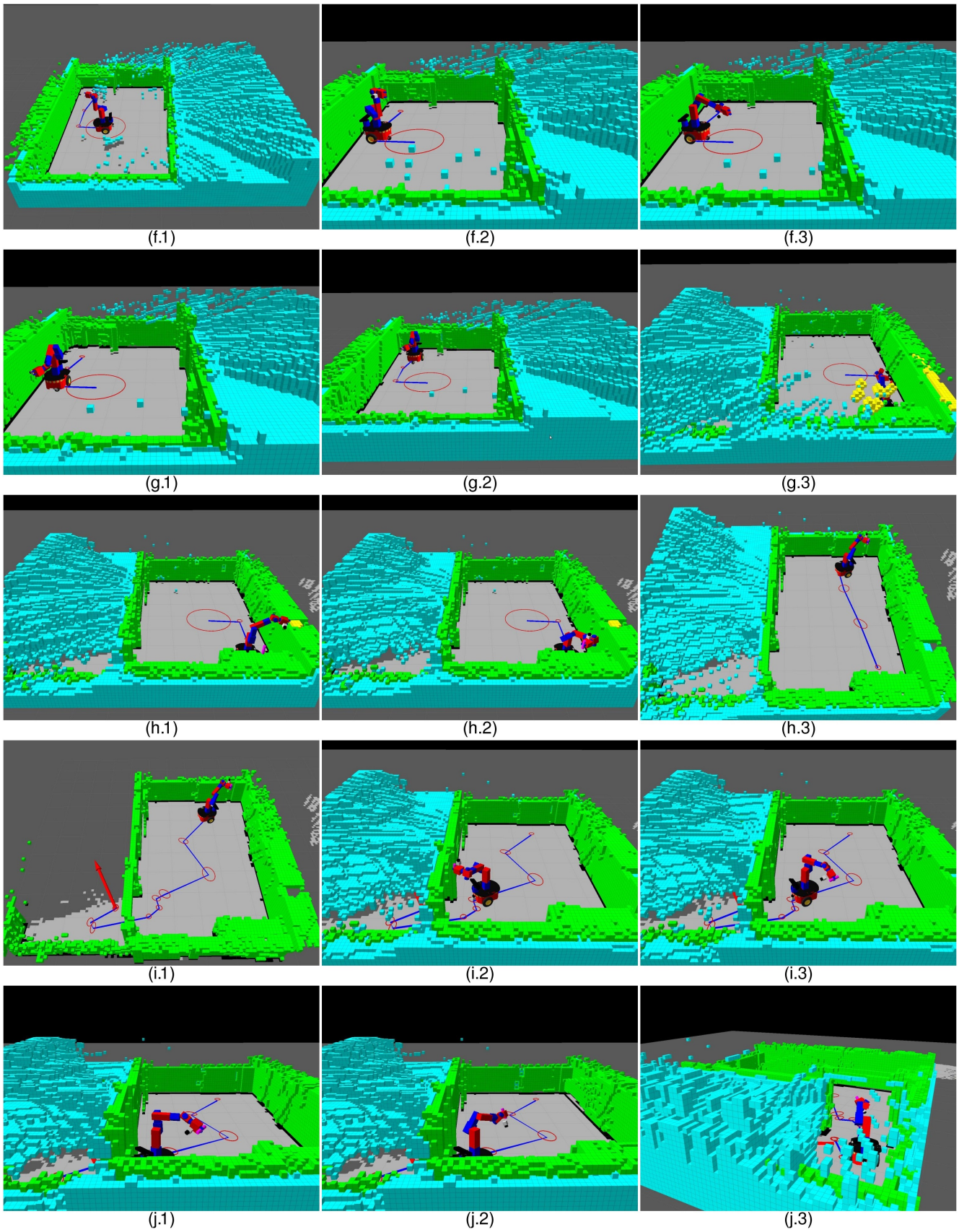

Fig. 11. Continued...

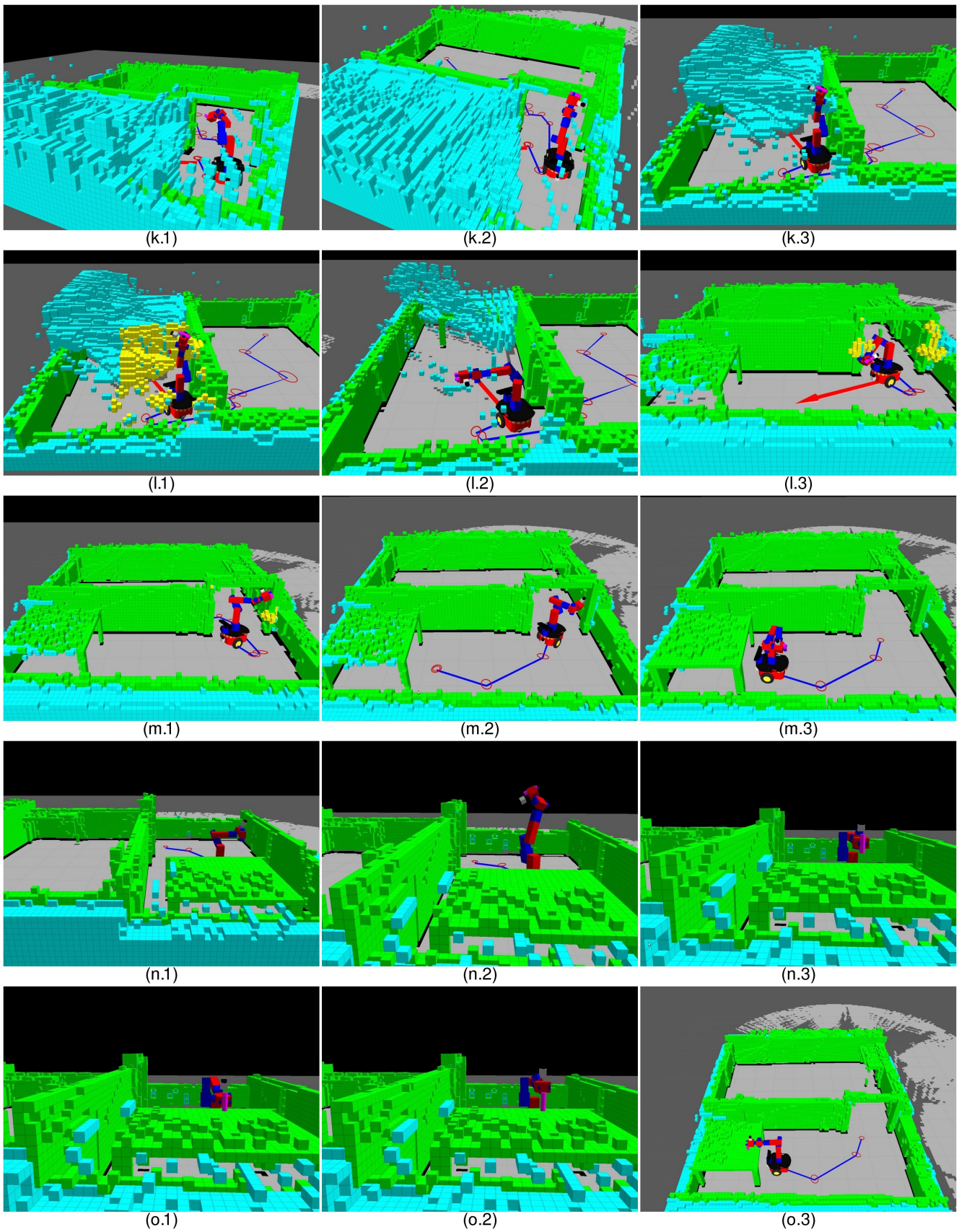

Fig. 12. Simulation test for pick-and-place task in unknown environment to demonstrate HAMP-BUA. (a.1) shows the initial unknown environment and (o.3) shows the environment after exploration along with object placement. Please see text for description.

TABLE IV

COMPARISON OF SIMULATIONS AND REAL EXPERIMENTS FOR PICK-AND-PLACE TASK IN UNKNOWN ENVIRONMENT.

Detail	Simulations	Real Experiments
Total trials	8	2
Total time taken (avg.)	120 minutes	150 minutes
Total execution time (avg.)	100 minutes	116 minutes
Total computational time (avg.)	20 minutes	34 minutes
Total Hokuyo scan time (avg.)	49 minutes	63 minutes
Num. of EXPLORE_A calls	3	4
Num. of NBV-A reached	18	23
Num. of EXPLORE_B calls	4	8
Num. of NBV-B reached	2	3

vironment and the outcomes are provided in Table IV. On an average our system took 2 hours to completely explore the environment and completes the pick-and-place task. However, it is important to note that only 17% of total time is the computational time while the remaining is motion execution time (100 minutes) that includes physically moving the mobile base or the arm, whether it be for executing plans for manipulator to reach NBVs-A and then scanning using eye-in-hand sensor, or for base to reach NBVs-B or for pick and place motions, etc. Of the 100 minutes, 49 minutes were taken by arm scanning motion execution at NBV-A[2] while arm motion execution to reach NBVs-A took 38 minutes. We believe that CPU is consumed by multiple tasks like simulation platform (Gazebo, ROS) visualization tool (Rviz, ROS) and hence slows down the arm motion. On an average, the system invoked EXPLORE_A 3 times and EXPLORE_B 4 times for a total of 18 NBV-A and 2 NBV-B were reached. Our EXPLORE_B calls also include call to reach pick base pose or place base pose.

One of the simulation trials is depicted from Figures 10 to 12 and also available in the attached video (Extension 1). In the figures, cyan colour represents the unknown regions, green colour represents the obstacles, magenta as Voxelmap and frontiers are represented by yellow colour. Screenshots in (a) show the unknown region (6m $\times$ 4m $\times$ 2m) in the beginning and the initial known region assumed around the mobile manipulator. The Voxelmap and frontiers at start are shown in (a.2). In the beginning, EXPLORE_A module was invoked to explore the local region and the screenshots from (b) to (e) show the 2D and 3D environment explored at different iterations of arm view planning. In total, it took 10 iterations, i.e., 10 NBVs-A were reached, and the region explored at the end is shown in (e.1) and (e.3). Within the known region, the pick base pose was reachable, therefore, a path was planned to reach as shown in (f.1). Screenshots from (f) to (g) show the execution of mobile manipulator path. Along the path, the arm reconfigured once as shown in (f.2) and (f.3). After reaching pick base pose, the grasping was not possible due to unexplored region around the vicinity of object, therefore, arm view planning was invoked to clear the unknown region. (g.3) and (h.1) show the frontiers before and after arm view planning. The object is grasped in (h.2). Post-grasping, the mobile manipulator moved toward already explored region (h.3) as there were few unknown voxels left. From there, a new NBV-B was searched and a path was planned (i.1). Screenshots from (i) to (k) show the path execution. Along the path, arm reconfigured 4 times, also shown in the screenshots. After reaching to NBV-B, arm view planning was invoked to explore local region. Frontiers at different iterations are shown from (i.1) to (m.1) (not all steps are shown). After local exploration (took 7 iterations), a path to reach place base pose was found and the object was placed as shown from (m.2) to (o.2). Finally explored environment is shown in (o.3).

[2]Recall that the Hokuyo line scan sensor is rotated using the last joint to get an area scan.

Fig. 13. Real environment for pick-and-place task. The mobile manipulator start configuration and object (bottle) are shown in the top figure while the bottom figure shows the table on the other side of the door where object should be placed.

B. Real experiments

For real experiments on SFU mobile manipulator, we used the environment shown in Figure 13 to demonstrate our integrated and autonomous system for pick-and-place task in unknown environment. We carried out two trials and the outcomes are provided in Table IV. As compared to simulations where the Hokuyo sensor was not sensitive to black surfaces, the system for real experiments took longer (150 minutes) to explore the environment, pick the object and place it at target location. This is because of three key reasons: a) we moved the arm with slow speed (maximum speed of 0.05 radians/seconds) for safety reasons due to some hardware issues with our Schunk arm at the time of experiments, b) many surfaces in our real environment (our lab) were black and for that Hokuyo does not work well,

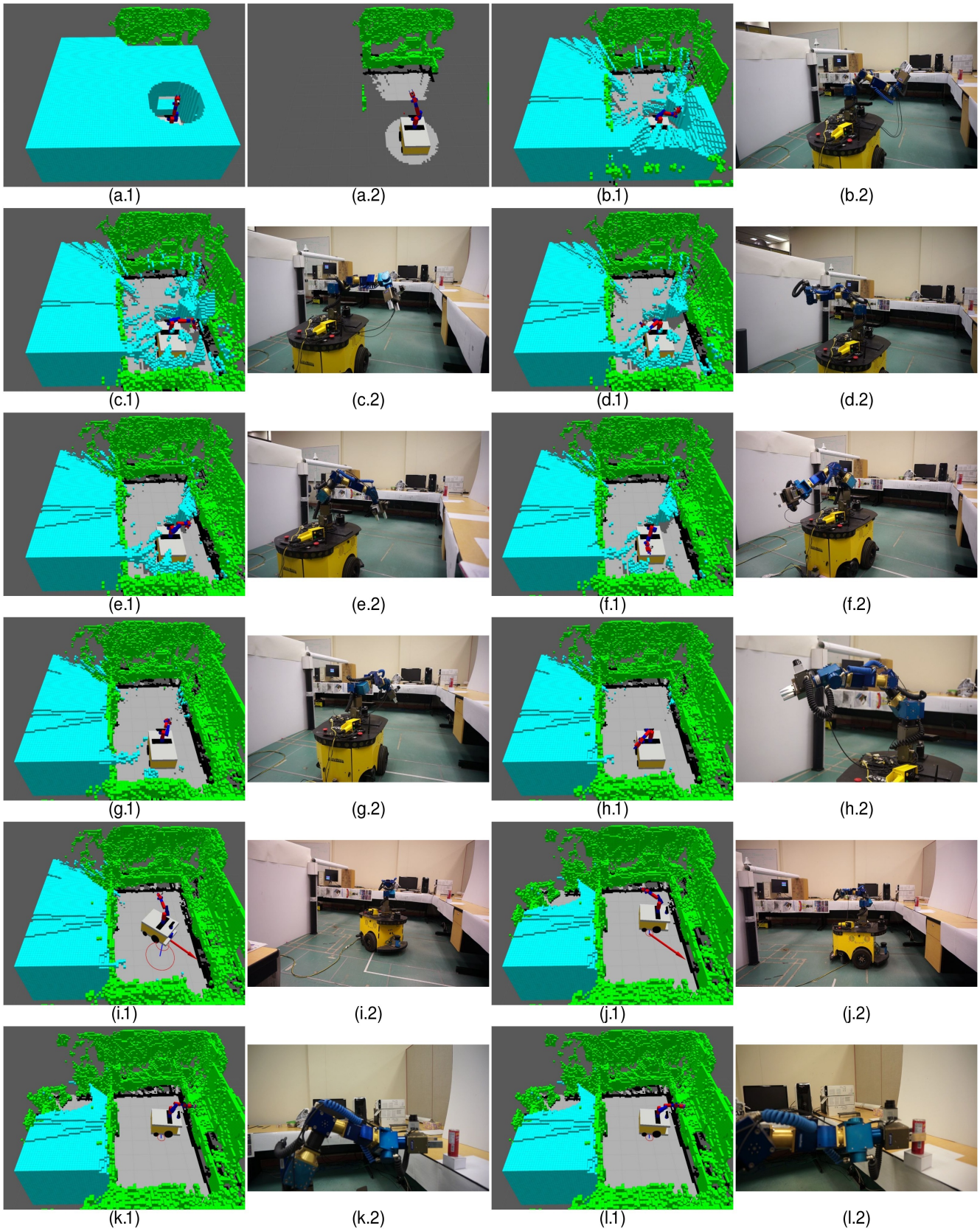

Fig. 14. Continued...

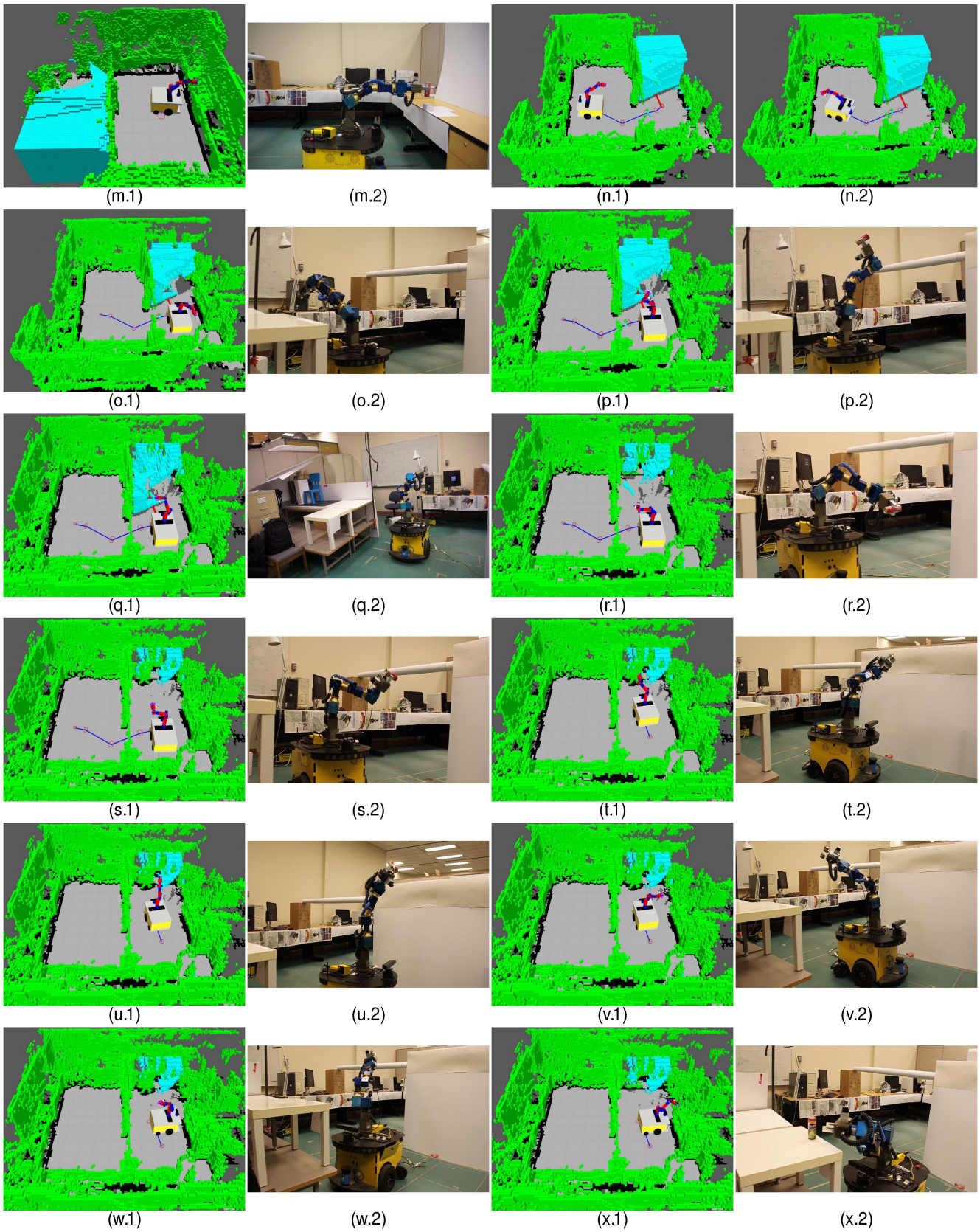

Fig. 15. Real experiment for pick-and-place task in unknown environment to demonstrate HAMP-BUA. (a.1) shows the initial unknown environment and (x.1) shows the environment after exploration. Please see text for description.

thereby, taking more NBVs-A iterations (Kneip et al., 2009), c) timing issues associated with insertion of Hokuyo and Kinect scans into global Octomap. Note that we hid some of the black surfaces by covering with papers as can be seen in the screenshots of the environment. Due to issues with eye-in-hand sensor (Hokuyo), the system took more number of iterations in a EXPLORE_A call, i.e., in total 23 NBVs-A were reached. Also, the system took 8 EXPLORE_B calls that include few of the failed attempts (3), for example, pick or place base pose was collision-free but the system failed to find a path with in the permitted time. This also shows that our system is robust to failure of individual modules as it tries to revisit the same problem next time in the loop.

One of the trials for real experiment is demonstrated from Figures 14 to 15. Screenshots in (a) show the unknown and known region in the beginning. The arm view planning was invoked at start base pose to explore the local region surrounding the mobile manipulator and screenshots from (b) to (h) show the eye-in-hand sensor at different NBVs-A and the environment left unexplored after each scanning from a NBV-A. This EXPLORE_A call took 15 iterations, i.e., 15 NBV-A were reached and the environment cleared at the end is shown in (h.1). Compared to simulation, the EXPLORE_A module in real experiment roughly takes 30% more time to explore the same amount of space. With in the explored region, the pick base pose was not reachable, therefore, a path was planned using HAMP-BUA to reach NBV-B shown in (i). Thereafter, the pick base pose was reached and the object was grasped as shown in (j) and (k)-(m), respectively. Screenshots from (n) to (o) show the mobile manipulator path execution to reach NBV-B to explore the unknown region on the other side of the door. From the reached NBV-B, the EXPLORE_A module was invoked that took 8 iterations to explore the local region as shown from (p) to (s). Note that, post arm exploration, there was some unexplored region left (s.1) but that was outside the local Voxelmap (not shown here) and on the other hand the place base pose was reachable with in the explored region. Therefore, a path was planned and figures from (t) to (v) show the arm reconfiguration step along the path. In (w) and (x), the mobile manipulator reached to place base pose and the object was placed at target location. Figure 16 shows the explored environment after completing the task. This real experiment trial is also shown in the video attached to this paper (Extension 2). Figure 17 shows the final outcome of our second trial where the environment was fully explored.

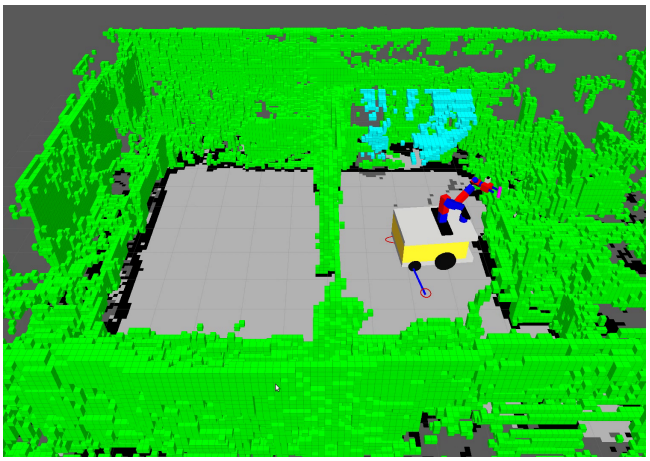

Fig. 16. [Real experiment trial 1]: less than 1% of the environment remained unexplored (in cyan colour) as the system was able to complete the pick-and-place task within the known region.

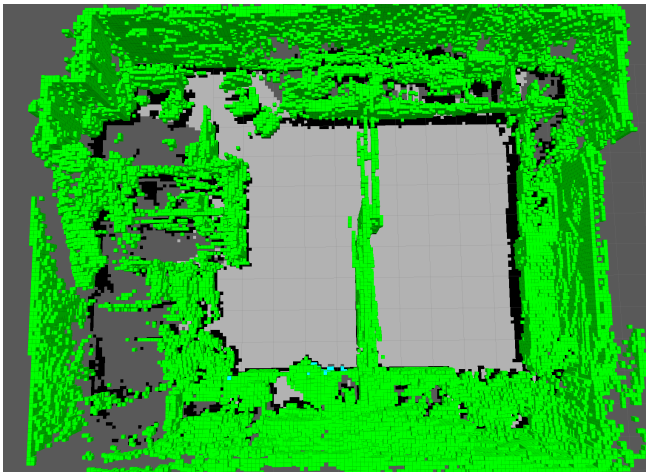

Fig. 17. [Real experiment trial 2]: fully explored environment.

VIII. Conclusion

We presented a sampling-based mobile manipulator planner (HAMP-BUA) that plans for both the base and the arm in a judicious manner and considers the base pose uncertainty and the effects of this uncertainty on manipulator motions. It uses localization aware sampling and connection strategies to consider only those nodes and edges which contribute toward better localization. This helps to reduce the planning time significantly which is needed for mobile manipulator where collision-checks are carried out in 3D. At path search stage, our planner incorporates base pose uncertainty along the edges (where arm remains static) and the effects of this uncertainty are considered on arm reconfiguration step at nodes (where base remains static). Moreover, HAMP-BUA respects the collision probability threshold along the path and uncertainty threshold at goal. First, we evaluated our planner in known environment and show that it finds a safer path as compared to other variants where uncertainty is not considered at different levels, for example, not incorporating base uncertainty on manipulator plans, not respecting collision probability threshold along the edges. We also show that variants of this planner that do not use our localization aware sampling and connection strategies will take longer to find the same quality of path.

Furthermore, we demonstrated our planner for a real world application using an integrated and fully autonomous system that carries out mobile pick-and-place tasks in unknown static environments. A key aspect of our integrated system is that the planner works in tandem with base and arm exploration modules (view planning) that explore the unknown

environment. Our system is demonstrated both in simulation and on the actual SFU mobile manipulator. The total time taken by the system (especially motion execution time) can be further reduced to great extent if the issues mentioned in Sections VII-A and VII-B can be resolved.

REFERENCES

Agha-mohammadi A, Chakravorty S and 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.

Bai H, Hsu D and Lee WS (2014) Integrated perception and planning in the continuous space: A POMDP approach. *The International Journal of Robotics Research* 33(9): 1288–1302.

Berenson D, Kuffner J and 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.

Bohlin R and Kavraki L (2000) Path planning using lazy prm. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*. pp. 521–528.

Bouilly B, Simeon T and 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.

Bry A and Roy N (2011) Rapidly-exploring random belief trees for motion planning under uncertainty. In: *Proc.of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. pp. 723–730.

Chitta S, Jones EG, Ciocarlie M and Hsiao K (2012) Mobile manipulation in unstructured environments. *IEEE Robotics and Automation Magazine* 19(2): 58–71.

Ciocarlie M, Hsiao K, Jones EG, Chitta S, Rusu RB and Sucan IA (2010) Towards reliable grasping and manipulation in household environments. In: *Intl. Symposium on Experimental Robotics*.

Collet A, Martinez M and Srinivasa SS (2011) The MOPED framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research* 30(10): 1284–1306.

Dogar M and Srinivasa SS (2010) Push-grasping with dexterous hands: Mechanics and a method. In: *Proc.of the IEEE International Conference on Intelligent Robots and Systems (IROS)*. pp. 2123–2130.

Dornhege C and Kleiner A (2011) A frontier-void-based approach for autonomous exploration in 3d. In: *Proc.of the IEEE International Symposium on Safety, Security, and Rescue Robotics*. pp. 351–356.

Dumitrescu I and Boland N (2002) Algorithms for the weight constrained shortest path problem. *International Transactions in Operational Research* 8(1): 15–29.

Eppner C, Höfer S, Jonschkowski R, Martin RM, Sieverling A, Wall V and Brock O (2016) Lessons from the amazon picking challenge: Four aspects of building robotic systems. In: *Robotics: Science and Systems (RSS)*.

Fraichard T and 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 and 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.

Hershberger J, Maxel M and Suri S (2007) Finding the k shortest simple paths: A new algorithm and its implementation. *ACM Transactions on Algorithms* 3(4).

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

Huang Y and 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 and 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 and Cassandra A (1998) Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101: 99–134.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (IJRR)* 30(7): 846–894.

Kneip L, Tche F, Caprari G and Siegwart R (2009) Characterization of the compact hokuyo URG-04LX 2D laser range scanner. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*. pp. 1447–1454.

Kragic D and Christensen H (2003) Robust visual servoing. *International Journal of Robotics Research* 22: 923–939.

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

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

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

Lazanas A and Latombe JC (1995) Motion planning with uncertainty: a landmark approach. *Artificial Intelligence* 76(1-2).

Leeper A, Hsiao K, Chu E and Salisbury JK (2010) Using near-field stereo vision for robotic grasping in cluttered environments. In: *Intl. Symposium on Experimental Robotics*.

Lehner P, Sieverling A and Brock O (2015) Incremental, sensor-based motion generation for mobile manipulators in unknown, dynamic environments. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*.

Marder-Eppstein E, Berger E, Foote T, Gerkey BP and 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 NA and Simmons R (2007) Particle RRT for path planning with uncertainty. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*. Roma, Italy, pp. 1617–1624.

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

Pilania V and Gupta K (2014) A hierarchical and adaptive mobile manipulator planner. In: *Proc. of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Madrid, Spain, pp. 45–51.

Pilania V and Gupta K (2015a) A hierarchical and adaptive mobile manipulator planner with base pose uncertainty. *Autonomous Robots* 39(1): 65–85.

Pilania V and Gupta K (2015b) A localization aware sampling strategy for motion planning under uncertainty. In: *Proc.of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 1187–1192.

Pilania V and Gupta K (2017) Localization aware sampling and connection strategies for incremental motion planning under uncertainty. *Autonomous Robots* 41(1): 111–132.

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

Prentice S and 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.

Scholz J, Chitta S, Marthi B and 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.

Shen S, Michael N and Kumar V (2012) Autonomous indoor 3d exploration with a micro-aerial vehicle. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*. pp. 9–15.

Stachniss C, Grisetti G and Burgard W (2005) Information gain-based exploration using rao-blackwellized particle filters. In: *Proc.of the Robotics: Science and Systems (RSS)*. pp. 65–72.

Tan J and 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 HG and Kyriakopoulos KJ (2000) Nonholonomic motion planning for mobile manipulators. In: *Proc.of the IEEE International Conference on Robotics and Automation (ICRA)*. pp. 1233–1238.

Torabi L (2011) *Integrated view and path planning for a fully autonomous mobile-manipulator system for 3D object modeling*. Ph.d. thesis, Simon Fraser University.

Torabi L and Gupta K (2012a) An autonomous 9-dof mobile-manipulator system for in situ 3d object modeling. In: *Proc.of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 4540–4541.

Torabi L and Gupta K (2012b) An autonomous six-dof eye-in-hand system for in-situ 3d object modeling. *International Journal of Robotics Research (IJRR)* 31(1): 82–100.

van den Berg J, Abbeel P and 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.

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

Yamauchi B (1997) A frontier-based approach for autonomous exploration. In: *Proc.of the Computational Intelligence in Robotics and Automation*. pp. 146–151.

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

Yu Y and Gupta K (2001) On eye-sensor based path planning for robots with non-trivial geometry/kinematics. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. pp. 265–270.

APPENDIX I
INDEX TO MULTIMEDIA EXTENSIONS

TABLE V
TABLE OF MULTIMEDIA EXTENSIONS

Extension	Type	Description
1	Video	HAMP-BUA demonstration using autonomous system for mobile pick-and-place task in unknown environment (simulation)
2	Video	HAMP-BUA demonstration using autonomous system for mobile pick-and-place task in unknown environment (real experiment on SFU Mobile Manipulator)