Autonomous System for Mobile Pick-and-Place Task in Unknown Environments

Vinay Pilania and Kamal Gupta

***Abstract*— We present a fully autonomous and integrated system that solves a critical and challenging problem in service robotics, that of end to end pick-and-place task (with and without task constraints) in unknown static environments. Our system is illustrated on a mobile manipulator equipped with base mounted and eye-in-hand sensors. The system intertwines one-step mobile manipulator planning and base mounted sensor scan with multi-steps manipulator planning interleaved with eye-in-hand sensor based exploration. We believe that our integrated pick-and-place system scores several firsts: a) its competency is far superior to that of the previous integrated planning systems in that it considers the currently unexplored (and unknown) region as obstacle, thereby resulting in completely safe paths, b) it explores the unknown environment as well as unknown grasping object using both base mounted and eye-in-hand sensors, c) it integrates two different view planning schemes to build a global world representation in term of Octomap, and finally d) the underlying planners judiciously generate safer paths for next best view of the base (NBV-B) and the arm (NBV-A) by considering base pose uncertainty and its effects on manipulator motions. We demonstrate our system both in simulation and on the actual SFU mobile manipulator. We also experimented with incorporated task constraints and report on lessons learned on system specifics and practical issues.**

I. Introduction

An end to end pick-and-place task in unknown environment is one of the challenging problems that need to be solved in order to successfully attain the deployment of fully autonomous robots in service and personal robotics like warehouses, homes, hospitals, etc. The solution to such problem first requires solving many other sub-problems: (i) navigating from one location to another in the known environment while considering uncertainty and task constraints (if any), (ii) exploring the unknown environment with the help of sensor(s) mounted on the robotic platform, (iii) pick-and-place pipeline to automatically grasp an object and place it at the desired location, (iv) maintaining global world representation to know which part of environment is known, unknown (unexplored) and obstacle, (v) estimating the pose of the robotic platform. The modules associated with these sub-problems need to be tightly integrated so that a unified framework can be obtained that is capable of carrying out end to end pick-and-place task in unknown environment. The robotic platform for implementing such unified framework could be many, for example, most popular ones are wheeled mobile manipulator and humanoid. Recently, the researchers also attempted to mount the manipulator on small sized unmanned aerial vehicle and underwater robotic boat.

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

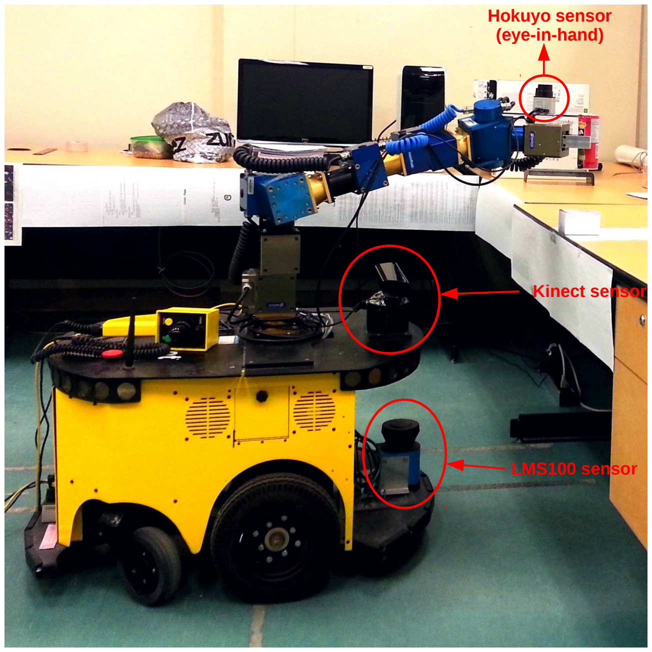


Fig. 1. The SFU mobile manipulator (Powerbot mobile base, Schunk 6 DOF powercube arm, Schunk two-finger parallel gripper) and mounting locations of different sensors.

The unknown environment could be either static or dynamic, i.e., containing moving obstacles. Consideration of moving obstacles brings more challenges mostly for the sub-problems (i) to (iv) as mentioned above. Therefore, we limit our proposed system to handle only the unknown static environments.

We propose a fully autonomous and integrated system that carries out an end to end pick-and-place task in unknown static environments and demonstrated the system using a wheeled mobile manipulator as shown in Figure 1. 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). 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. We assume that very small region around mobile manipulator is known in the beginning but other than that the entire environment is unknown. We believe that our integrated pick-

and-place system scores several firsts: a) its competency is far superior to that of the previous integrated planning systems in that it considers the currently unexplored (and unknown) region as obstacle, thereby resulting in completely safe paths, b) it explores the unknown environment as well as unknown grasping object using both base mounted and eye-in-hand sensors, c) it combines two different view planning schemes to build a global world representation in term of Octomap, and finally d) the underlying planners judiciously generate safer paths for next best view of the base (NBV-B) and the arm (NBV-A) by considering base pose uncertainty and its effects on manipulator motions.

Please note that our integrated system was partly mentioned in [1], however, that was in limited sense and mainly in the context to show the competence of our HAMP-BUA algorithm (Hierarchical and Adaptive Mobile Manipulator Planner with Base pose Uncertainty and its propagation to Arm motions) [1]. Here, we present it as a standalone work with more detailed information regarding system implementation. For example, we provide the implementation detail of local Voxelmap and view planning for the arm, both are inter-connected and important components of the integrated system. The local Voxelmap is used by the arm view planning to compute next best view NBV-A. Key parameters associated with arm view planning are also discussed. Moreover, we bring out the details of practical issues faced during real implementation like getting incomplete scan from Hokuyo eye-in-hand sensor due to black surfaces, not all sensor scans are inserted into global world representation (Octomap) due to high frequency of incoming scans and limited capacity of Octomap in term of scan insertion. The missing scans and the corresponding remedy lead to serious repercussions on the computational and execution time to complete the entire task. It is straightforward to extend our system to incorporate task constraints, i.e., end to end pick-and-place task can be carried out in unknown static environment while maintaning task constraints. This is achievable if the planners used in arm and base view planning are replaced with their task variants. The task constraint version of HAMP-BUA is available in [2]. Based on our experimental results, it is not that easy in practice (however, theoretically seems feasible) to carry out arm view planning while taking into account the task constaints using a robotic platform similar to ours. We also discuss our results for the same. Note that end to end pick-and-place task with task constaints in unknown static environment is not a crazy idea. We see that sort of scenarios in our day to day routines, for example, a person is holding a coffee mug and running to catch the train while navigating through the crowd (unknown environment) without switching the mug from one hand to another as well as keeping the mug upright (maintaining constraint) in order to avoid spilling.

II. Related Work

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 [3]. 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 [4]. 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. [5] 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 [6]. 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 [7] 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. There is lot of work related to environment exploration and mapping both in 2D and 3D either using mobile base [8] or UAVs [9]. However, we have not come across any work where mobile manipulator is used to explore the unknown environment and achieve some tasks (for example, end to end pick-and-place).

III. Robotic Platform

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 1.

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.

IV. Autonomous System for Mobile Pick-and-Place Task in Unknown Environments

In this section, we report our integrated and autonomous system for end to end pick-and-place tasks in unknown static environments.

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.

A. System description

Our integrated and autonomous system architecture is described in Figure 2. 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 [3] 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.

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

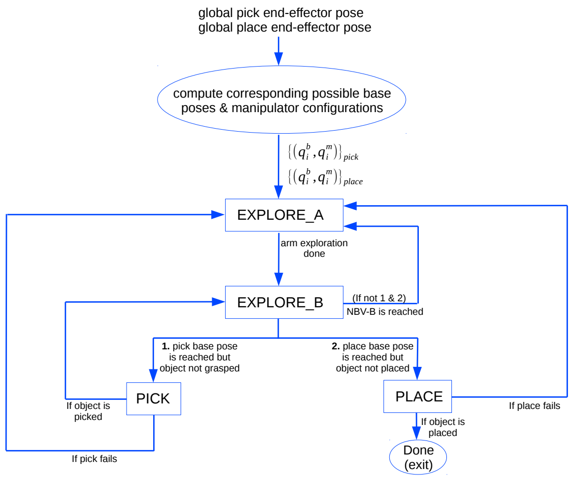

global pick end-effector pose
global place end-effector pose
compute corresponding possible base poses & manipulator configurations
$\{(q_i^b, q_i^m)\}_{pick}$
$\{(q_i^b, q_i^m)\}_{place}$
EXPLORE_A
arm exploration done
EXPLORE_B
(If not 1 & 2)
NBV-B is reached
1. pick base pose is reached but object not grasped
2. place base pose is reached but object not placed
PICK
If object is picked
If pick fails
PLACE
If place fails
If object is placed
Done (exit)

Fig. 2. An integrated and autonomous system for end to end pick-and-place task in unknown environments.

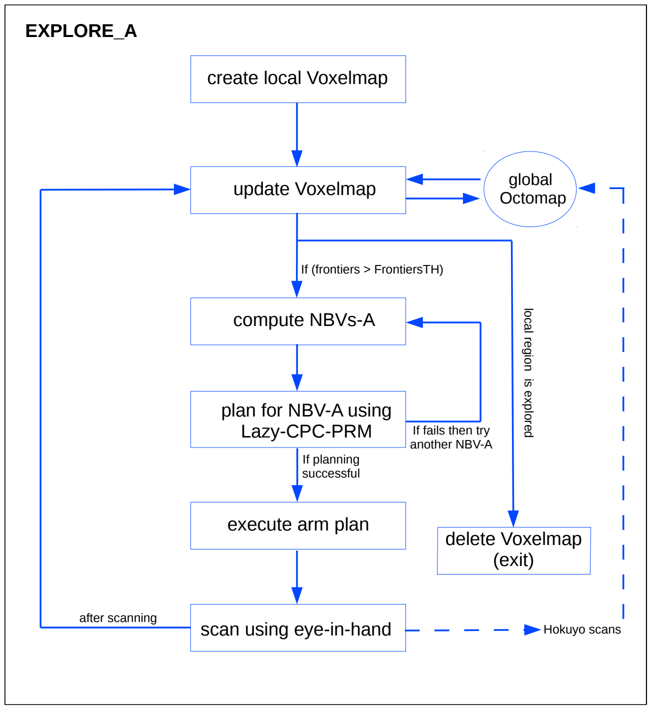


Fig. 3. EXPLORE_A module of the system.

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 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

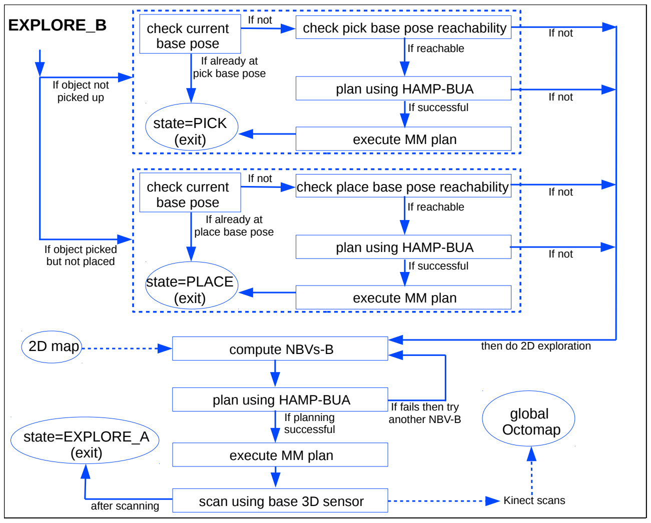


Fig. 4. EXPLORE_B module of the system.

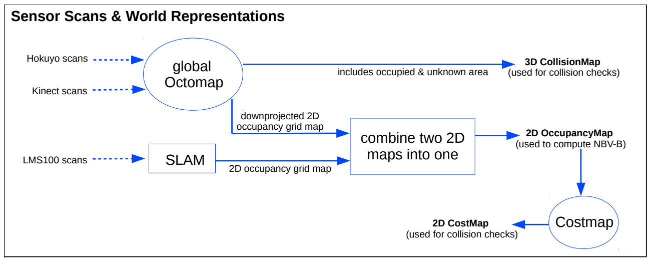

Sensor Scans & World Representations
Hokuyo scans
Kinect scans
global Octomap
includes occupied & unknown area
3D CollisionMap
(used for collision checks)
downprojected 2D occupancy grid map
combine two 2D maps into one
2D OccupancyMap
(used to compute NBV-B)
LMS100 scans
SLAM
2D occupancy grid map
2D CostMap
(used for collision checks)
Costmap

Fig. 5. Sensor scans and world representations (2D and 3D).

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 3 to 4. We now describe these modules.

EXPLORE_A (see Figure 3) creates a local Voxelmap, a 3D version of occupancy grid map, 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 (3D world representation). 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. 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 sensor. Note that scans are inserted into the global Octomap and not the local Voxelmap. Lazy-CPC-PRM [10] 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 4) 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 [11]. The base view planning uses a 2D occupancy grid map to compute NBV-B and this map comes from a series of steps as shown in Figure 5.

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 [12] 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 [12], [13]. 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 [14]. For cases where the object and gripper can not be visually tracked, methods such as [13], [15] can be incorporated. While in the presence of object pose uncertainty and high clutter, push-grasping approach in [16] can be helpful.

B. *Sensor scans and world representations*

Figure 5 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 [17], 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 by the planners as mentioned in Section IV-C.

C. *Planners for NBV-B and NBV-A*

In end to end pick-and-place task, there are mainly two types of movement (excluding gripper and rotating last joint of arm to form an area scan using eye-in-hand sensor): one is of the entire mobile manipulator to reach a NBV-B while the other is just manipulator's motion to reach the NBVs-A. For the former, it is possible to fold the arm to some safe configuration and just move the mobile base rather than moving the entire mobile manipulator (either sequential or continuous base and arm motions). However, there are scenarios where it is not possible to move from start base pose to goal base pose without reconfiguring the arm, see [18]. Previously, we designed a range of judicious sampling-based mobile manipulator planners that move the arm on a need to basis and consider uncertainty at different levels. HAMP is the deterministic version (assuming robot state is known), HAMP-U [19] considers only base pose uncertainty. HAMP-BUA [1] is the advance version that incorporates uncertainty at different levels, for example, uses localization aware sampling [20] and connection [21] strategies to eliminate nodes and edges that do not contribute toward better localization, considers base pose uncertainty and its propagation to arm motions. Therefore, we use HAMP-BUA as our core planner to generate motions for the mobile manipulator.

We employ Lazy-CPC-PRM [10] to generate manipulator motions which are needed to reach NBVs-A and for pick-and-place. This planner assumes static base position and computes manipulator plan by considering base pose uncertainty.

V. Independent Evaluation of Arm View Planning Module

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. We carried out 40 trials and for each trial (with and without task constraint) 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. The task is to explore the unknown region within the boundaries of a local Voxelmap. Our implementation is in C++ under linux and runs on a Pentium dual core 2.5 Ghz computer with 4GB memory.

A. *Local Voxelmap and Arm View Planning Parameters*

In EXPLORE_A, we use three parameters. Two of them are an integral part of traditional information-based arm view planning: the maximum number of iterations to try before declaring that arm view planning is over even if it is not (we use 15), the maximum number of valid samples (sensor views) to be searched to find the next best view (we use 50). While the third parameter (FrontiersTH as mentioned in Figure 3) is added by us that tells whether arm view planning is needed or not (at fixed base pose). If the number of frontiers in the local Voxelmap are below 10, the threshold value we used, then EXPLORE_A module declares that local region around mobile manipulator is already explored and arm view planning is not required at current base pose. Below we also explain the rational behind selecting FrontiersTH as 10 and not 0. The end-effector in our SFU mobile manipulator as shown in Figure 1 can maximally extend up to 0.76 m. Therefore, in EXPLORE_A, we construct a Voxelmap that contains 20700 voxel cells, each cell is a square of size 0.05 m. To give more insight, the Voxelmap contains 23 levels such that each level consists of 900 cells. Figure 6 shows the visualization of a Voxelmap (light magenta) where yellow colour denotes the frontiers. Initially, we assume that the region (cylinder of radius 0.5 m) surrounding mobile manipulator is known that is why the center region of Voxelmap is empty, i.e., known.

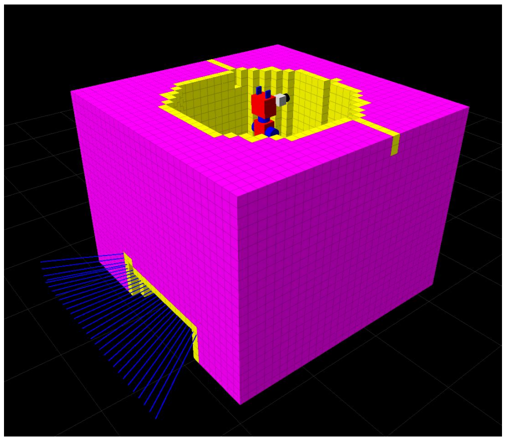


Fig. 6. Voxelmap with known region in the center. Voxel cells (unknown) are shown in light magenta and frontiers in yellow colour. The blue colour lines show the ray tracing model to simulate the eye-in-hand sensor to compute the information gain at NBV-A.

B. *With and without task constraints*

For experiments without any task constraint, we set 10 seconds as maximum permitted time for Lazy-CPC-PRM to plan a path while for experiments with task constraint, we set 60 seconds for Lazy-CPC-PRM$_{\text{TS}}$ [3]. For both the experiments, we 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 I (without constraint) and for the simplest scenario (no obstacles in surrounding) in Table II (with constraint). From Table I, 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. Sometime, the update of Voxelmap takes longer (0.7 second). We use service method of ROS (robot operating system) to communicate with global Octomap in order to know the status of voxel cells. The delay in updates relates to the computational load on Octomap, for example, it needs to insert scans and at the same time a request is made to update the Voxelmap. In our 40 trials, EXPLORE_A without task space constraint succeeded in

TABLE I
EXPLORE_A MODULE TEST IN CLUTTERED ENVIRONMENT WITHOUT TASK CONSTRAINTS.

	# 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	-	-

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 II
EXPLORE_A MODULE TEST IN SIMPLEST ENVIRONMENT WITH TASK CONSTRAINTS.

	# frontiers	Voxelmap update T. (sec)	NBV T. (sec)	Planning T. (sec)
1	1498	0.3418	2.91	60 (failed)
	-	-	-	16.18
2	1754	0.6897	3.32	60 (failed)
	-	-	-	37.78
3	1696	0.0376	3.18	28.35
4	1581	0.0078	3.04	12.17
5	1568	0.0321	3.99	60 (failed)
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
15	1490	0.3784	4.81	60 (failed)

On the other hand, EXPLORE_A with task space constraint failed to explore the Voxelmap even in single trial (15 iterations) as shown in Table II. Figure 7 shows the region remained unexplored even after 15 iterations. Seeing this, we increased the permitted time to plan a path to 180 seconds and maximum number of iterations to 25. Still the behaviour of EXPLORE_A with task space constraint remains same. We attribute the failure to two things: one is the Lazy-CPC-PRM$_{\text{TS}}$ or in general one can say a task space planner even without base pose uncertainty, while the second is the constraints put on end-effector as result of which the eye-in-hand scan explores less area as compared to when there are no constraints. Putting Kinect as eye-in-hand sensor has its own problems like it can not see upto 1 meter and then there are issues with the bandwidth of sensor data which we mention in Section VI. Because of ineffectiveness of EXPLORE_A with task constraint, for now, we limit our end to end pick-and-place task without using any task constraints and will pursue it separately in the future.

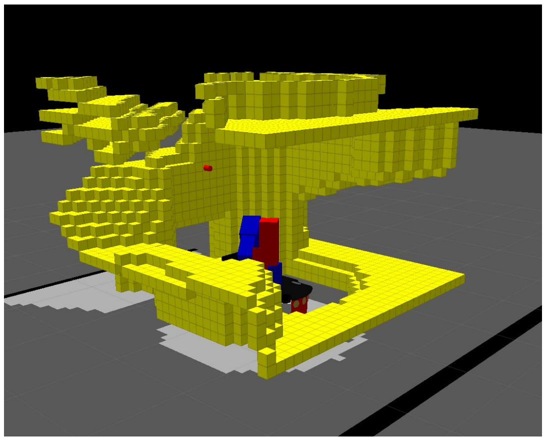


Fig. 7. EXPLORE_A with task space constraint failed to explore complete region of local Voxelmap. Only frontiers are shown in yellow colour, voxel cells are not shown in this screenshot. Red colour dot shows NBV-A but Lazy-CPC-PRMTs failed to find a path for it.

VI. Practical Issues: Incomplete Hokuyo Scans and Missing Scans in Octomap

For our experiments, we used Fuerte version of ROS. Therefore, the efficiency issues of Octomap that we mention here are related to that version. We believe that Octomap in recent ROS version might have been improved to some extent at least. In simulation, scans from Hokuyo sensor are published at the rate of 10 Hz and each scan consists of 683 points. While Hokuyo on real robot also publishes at the same rate but each scan consists of approximately 383 points. However, given the sensor view ranging from -1.57 to 1.56 at angle increment of 0.0061 radians (slightly greater than simulation which is 0.0046) there should be 514 points. That implies around 130 points are not reported by sensor as the reading of corresponding rays is zero. We found out that lot of surfaces in our real environment (our lab) are black and for that Hokuyo does not work well [22]. As we will see in the result that the exploration in real environment takes longer as compared to simulation. This is because the voxels covered by the rays that we are missing in case of real Hokuyo will not be cleared. For Kinect, we throttled the point cloud data and then downsampled to get scans at the reduced rate of 2 Hz (original was 30 Hz) such that each scan consists of around 11189 points (17500 without downsample). The job of global Octomap is to insert scans coming from Hokuyo and Kinect and also to publish 3D collision map as well as downprojected 2D map upto a certain height (we use 1 meter) as shown in Figure 5. Insertion of a Kinect scan takes 0.1 to 0.2 second and 0.01 to 0.02 second for a Hokuyo scan. While the publishing of maps (both occupancy and collision) takes approximately 1 second. Therefore, it is obvious that some Hokuyo and Kinect scans will be missed given the rate at which data is published and the rate at which global Octomap incorporates them. Our observation from experiments tells that Kinect contribution is only 25% in the exploration. Hokuyo is the main sensor that explores most of the environment. Therefore, one way to deal with missing scan issue is to reduce the speed at which Hokuyo takes the scans (we move arm with maximum speed of 0.05 radians/seconds).

TABLE III
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

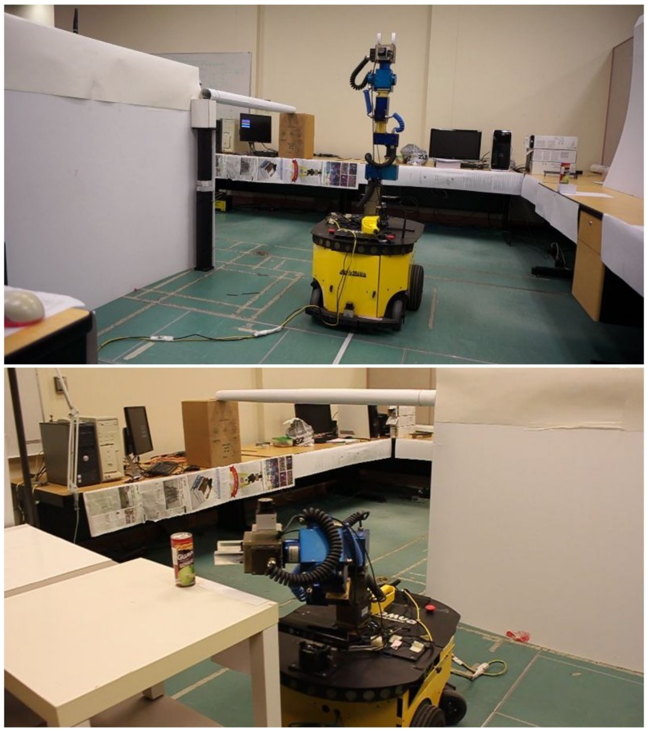
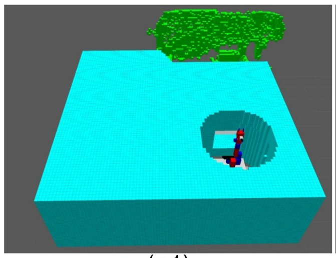
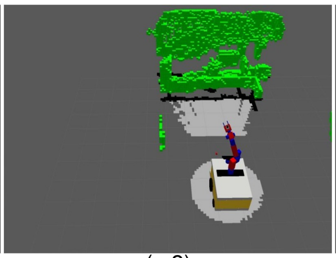
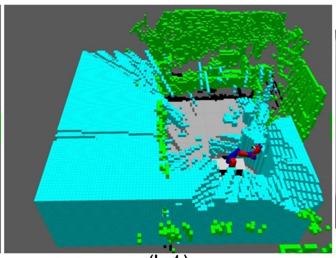
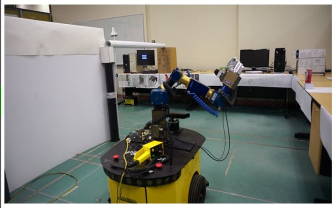


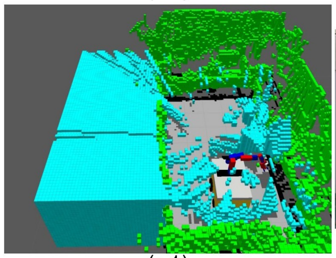
Fig. 8. Real environment for pick-and-place task. The mobile manipulator start configuration and object (bottle) are shown in the top figure while the bottom figure shows the table on the other side of the door where object should be placed.

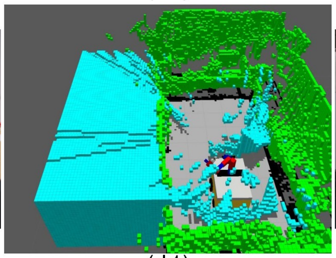
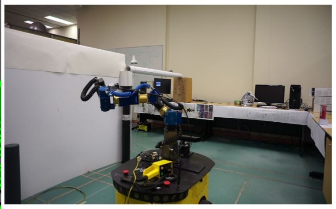
VII. Results

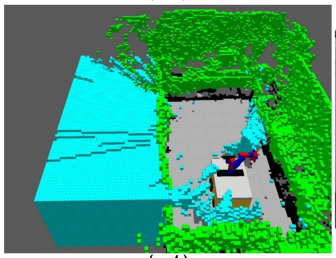
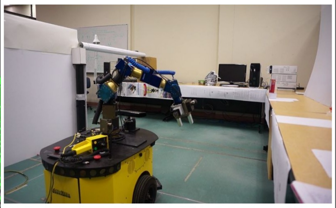
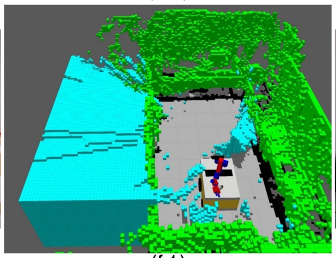
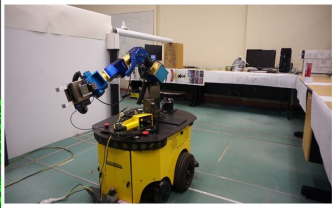
We ran our integrated and autonomous system in the simulation and real environments and the outcomes are provided in Table III.

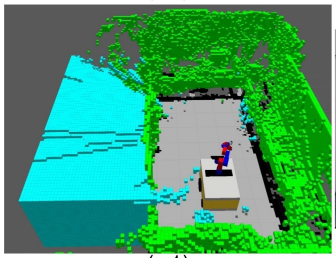
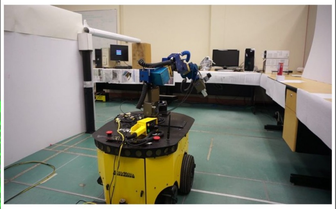
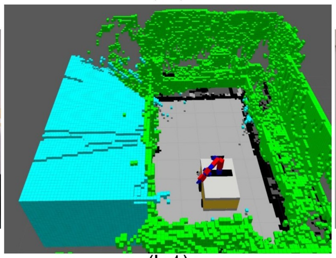
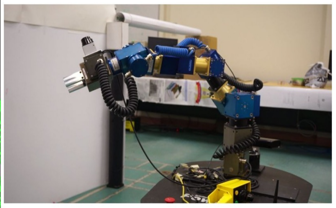
For simulation, on an average our system took 2 hours to completely explore the environment and completes the end to end pick-and-place task. However, it is important to note that only 17% of total time is the computational time while the remaining is motion execution time (100 minutes) that includes physically moving the mobile base or the arm, whether it be for executing plans for manipulator to reach

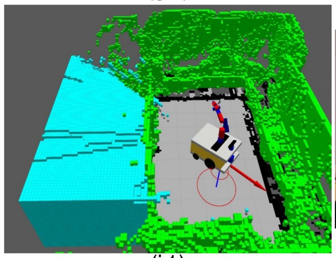
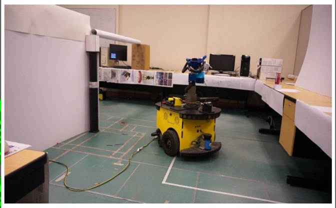
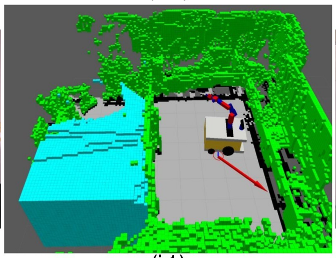

(a.1)

(a.2)

(b.1)

(b.2)


(c.1)

(c.2)

(d.1)

(d.2)


(e.1)

(e.2)

(f.1)

(f.2)


(g.1)

(g.2)

(h.1)

(h.2)


(i.1)

(i.2)

(j.1)

(j.2)

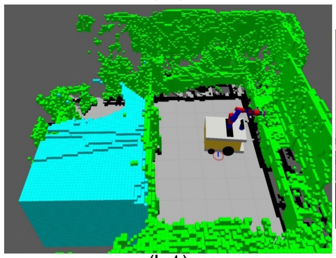
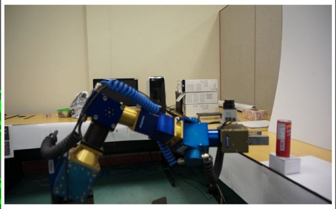
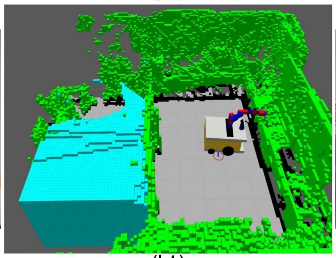
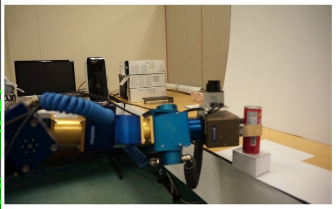
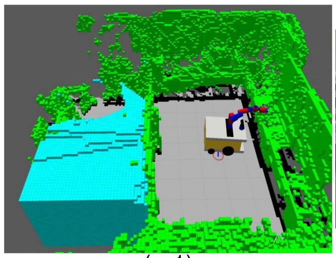
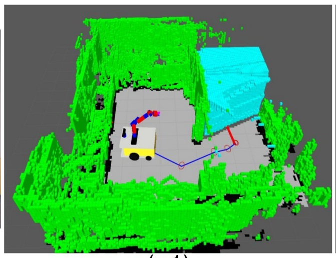
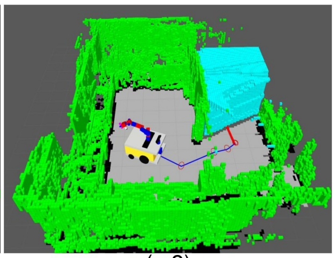
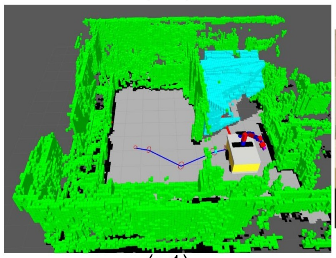
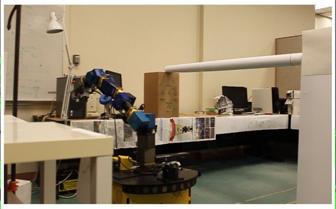
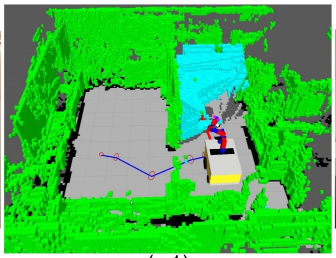
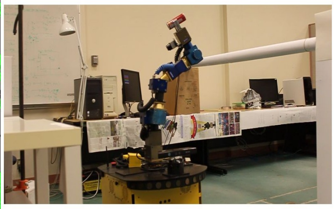
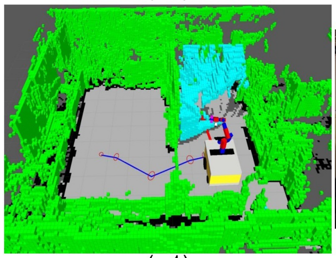
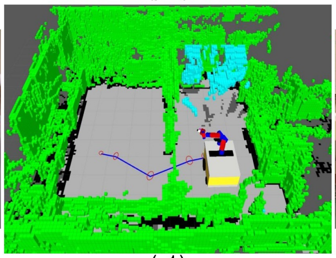
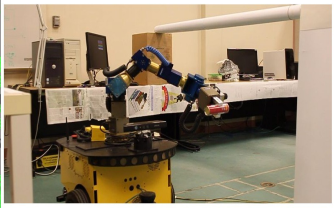
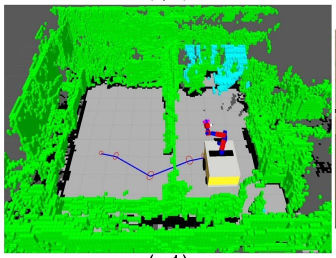
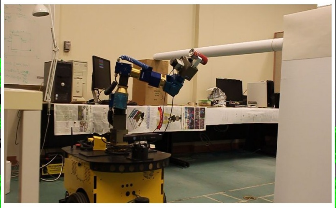
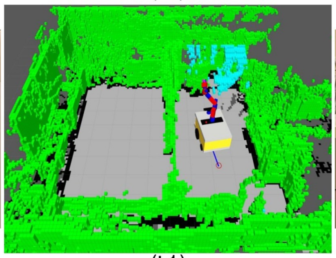
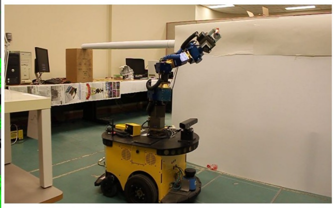
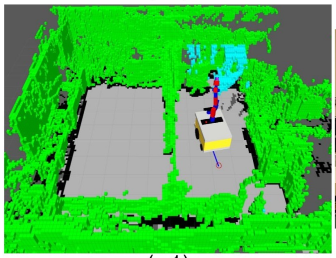

(k.1)

(k.2)

(l.1)

(l.2)

Fig. 9. Continued...


(m.1)

(m.2)

(n.1)

(n.2)

(o.1)

(o.2)

(p.1)

(p.2)

(q.1)

(q.2)

(r.1)

(r.2)

(s.1)

(s.2)

(t.1)

(t.2)

(u.1)

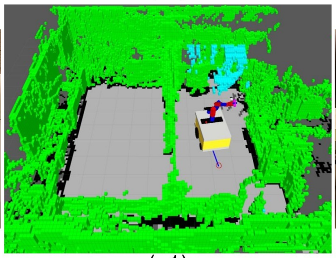
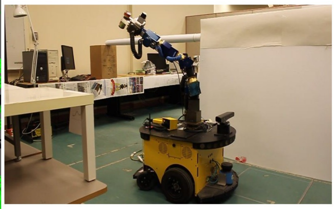
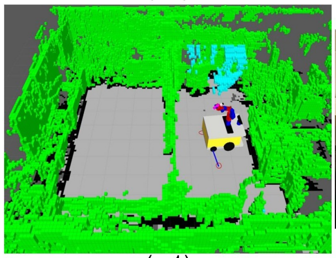
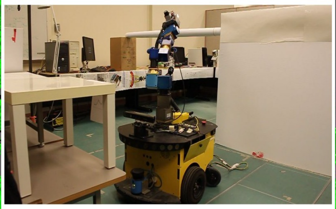
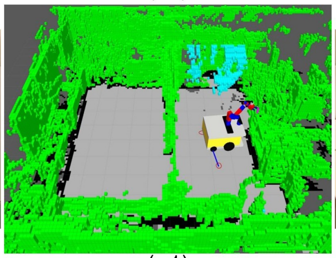
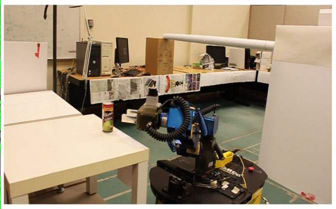
(u.2)

(v.1)

(v.2)

(w.1)

(w.2)

(x.1)

(x.2)

Fig. 10. One of the real experiments for end to end pick-and-place task in unknown environment. (a.1) shows the initial unknown environment and (x.1) shows the environment after exploration. Please see text for description.

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[1] 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 to a great extent. 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.

For real experiments on SFU mobile manipulator, we used the environment shown in Figure 8 to demonstrate our integrated and autonomous system. We carried out two trials and the outcomes are provided in Table III. 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, thereby, taking more NBVs-A iterations [22], 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 9 to 10. 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.

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

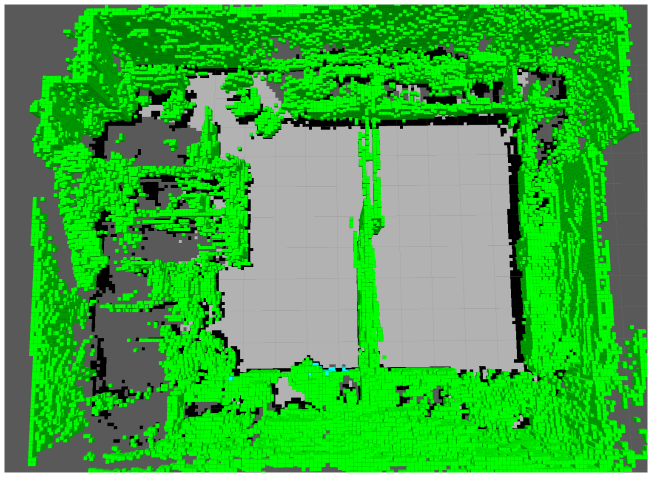


Fig. 11. [Real experiment trial 2]: fully explored environment.

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. This real experiment trial is also shown in the video attached to this paper. Figure 11 shows the final outcome of our second trial where the environment was fully explored.

VIII. Conclusion

We presented a fully integrated and autonomous system that carries out an end to end pick-and-place task in unknown static environments, a critical and challenging problem in service robotics. The system is demonstrated with the help of simulations and real experiments on SFU mobile manipulator. 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. The system presents unique way of "How it explores (combines two different exploration schemes into one)" and "How it maintains (global world representation - Octomap) and reuse the information for local purpose (local Voxelmap and 2D Occupancy map for arm and base view planning, respectively)". Moreover, it considers uncertainty associated with sensors and controls with the help of underlying mobile manipulator and manipulator planners, thereby providing safer plans for execution.

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 can be resolved. In future, we seek to improve the efficiency of our system and incorporate a more systematic pick-and-place pipeline. We also would like to explore the possibility of making the system work with task constraints.

References

[1] V. Pilania and K. Gupta, "Mobile manipulator planning under uncertainty in unknown environments," *The International Journal of Robotics Research*, vol. 37, no. 2-3, pp. 316–339, March 2018.

[2] V. Pilania, "Safe motion planning under uncertainty for mobile manipulators in unknown environments," Ph.D. thesis, Simon Fraser University, 2015.

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

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

[5] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3d," in *Proc.of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, Nov 2011, pp. 351–356.

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

[7] L. Torabi and K. Gupta, "An autonomous six-dof eye-in-hand system for in-situ 3d object modeling," *International Journal of Robotics Research (IJRR)*, vol. 31, no. 1, pp. 82–100, January 2012.

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

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

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

[11] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc.of the Computational Intelligence in Robotics and Automation*, 1997, pp. 146–151.

[12] A. Leeper, K. Hsiao, E. Chu, and J. K. Salisbury, "Using near-field stereo vision for robotic grasping in cluttered environments," in *Intl. Symposium on Experimental Robotics*, 2010.

[13] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Sucan, "Towards reliable grasping and manipulation in household environments," in *Intl. Symposium on Experimental Robotics*, Dec. 2010.

[14] D. Kragic and H. Christensen, "Robust visual servoing," *International Journal of Robotics Research*, vol. 22, pp. 923–939, 2003.

[15] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[16] M. Dogar and S. S. Srinivasa., "Push-grasping with dexterous hands: Mechanics and a method," in *Proc.of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 2123–2130.

[17] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, April 2013.

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

[19] ——, "A hierarchical and adaptive mobile manipulator planner with base pose uncertainty," *Autonomous Robots*, vol. 39, no. 1, pp. 65–85, June 2015.

[20] ——, "A localization aware sampling strategy for motion planning under uncertainty," in *Proc.of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015, pp. 1187–1192.

[21] ——, "Localization aware sampling and connection strategies for incremental motion planning under uncertainty," *Autonomous Robots.*, vol. 41, no. 1, pp. 111–132, 2017.

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