# Navigation in Three-Dimensional Cluttered Environments for Mobile Manipulation

Armin Hornung   Mike Phillips   E. Gil Jones   Maren Bennewitz   Maxim Likhachev   Sachin Chitta

***Abstract*— Collision-free navigation in cluttered environments is essential for any mobile manipulation system. Traditional navigation systems have relied on a 2D grid map projected from a 3D representation for efficiency. This approach, however, prevents navigation close to objects in situations where projected 3D configurations are in collision within the 2D grid map even if actually no collision occurs in the 3D environment. Accordingly, when using such a 2D representation for planning paths of a mobile manipulation robot, the number of planning problems which can be solved is limited and suboptimal robot paths may result. We present a fast, integrated approach to solve path planning in 3D using a combination of an efficient octree-based representation of the 3D world and an anytime search-based motion planner. Our approach utilizes a combination of multi-layered 2D and 3D representations to improve planning speed, allowing the generation of almost real-time plans with bounded sub-optimality. We present extensive experimental results with the two-armed mobile manipulation robot PR2 carrying large objects in a highly cluttered environment. Using our approach, the robot is able to efficiently plan and execute trajectories while transporting objects, thereby often moving through demanding, narrow passageways.**

## I. Introduction

Home assistance is a major future area of interest for personal robots. Navigation in the highly unstructured and dynamic environments of a home is thereby the prerequisite to fulfill any high-level tasks. Current state of the art navigation approaches enable fast planning and robust execution for a 3-DoF robot base in a 2D grid map indoors [1]. In these approaches, sensor data is either two-dimensional or projected down to 2D from 3D and a projected footprint is typically used to plan for the robot on a 2D grid map. However, these approaches are usually not able to generate motions to goal locations in highly cluttered areas or to goal locations where the 2D projection of the robot is in collision in the 2D map – even if the 3D configuration of the robot is actually collision-free in the 3D world. As an example, Fig. 1 shows the PR2 attempting to pick up and move a laundry basket off a table. It can only achieve this task by moving its base underneath the table with its arms over the table. Such a configuration is unachievable using traditional navigation planning techniques. Additionally, no path for the PR2 carrying the laundry basket might be found for narrow passages when planning motions in a 2D map. This is due

A. Hornung and M. Bennewitz are with the Humanoid Robots Lab, University of Freiburg, Germany. M. Phillips and M. Likhachev are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, USA. S. Chitta and E.G. Jones are with Willow Garage Inc., Menlo Park, USA.

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8, by the European Commission under grant agreement numbers FP7-248258-First-MM, and by Willow Garage Inc.

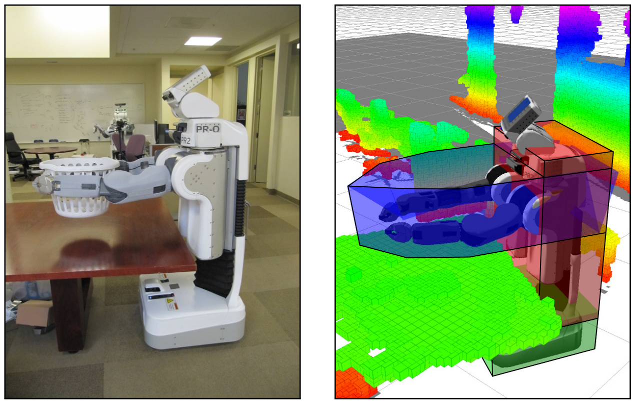

Fig. 1. *Left:* A mobile manipulation robot often has to approach obstacles closely when traversing narrow passages or picking up large objects. *Right:* To allow for efficient collision checks while considering the 3D structure of the environment and the robot, we use a multi-layered representation for the robot. Here, it consists of projected layers for the base (green), spine (red), and arms (blue) in addition to a full 3D collision map.

to the enlarged footprint in the 2D map needed to account for the basket and the extended arms.

Full 3D collision checking is a possible solution in such situations. In typical indoor environments with a multi-DoF robot, however, this is expensive and will lead to long planning times. Collision checking could be sped up using a coarser representation of the environment. However, the lack of high-resolution information can prevent the robot from reaching goals in clutter.

To overcome these limitations, we propose an integrated navigation framework that utilizes a combination of full 3D collision checking with a multi-layered 2D representation of both the robot and the environment (Fig. 1). Our approach starts with the robot incrementally building a 3D occupancy map in an efficient octree-based representation. This 3D map is then projected down into a multi-layered 2D representation. Based on the current configuration of the robot and any object it might be holding, corresponding projected footprints for the different layers are automatically computed. The advantage of this method is that expensive, 3D collision checks during planning can be avoided, when no collision is found in any layer. We apply a global planner to generate paths on a lattice graph based on the layered representation in an anytime manner. A local planner then executes this path and validates it during execution.

Our novel contributions are multi-fold. First, we present a system that is capable of efficient realtime 3D planning and navigation. Our approach is particularly suitable for mobile manipulation tasks, e.g., tasks where a robot needs to carry large objects in cluttered environments. Another important contribution is the integration of an efficient 3D representation for large-scale indoor environments with fast realtime motion planning, implemented and validated on a real robot operating with noisy sensor data in a cluttered environment. Our approach provides the first implementation of efficient navigation planning for mobile manipulation systems of different shapes in arbitrary configurations.

We validate our approach through real-world experiments on the PR2 mobile manipulation robot. Using our approach, the PR2 is able to carry large objects in a cluttered indoor environment. It is able to navigate collision-free and efficiently, transporting objects to parts of the workspace unreachable with traditional 2D navigation planning approaches. Our framework builds upon and extends the capabilities of the Search-based Planning Library (SBPL, [2]) and is available as open source software[1].

## II. Related Work

Navigation in cluttered environments is a well-studied problem for mobile robots [1], [3], [4], [5]. Most approaches to this problem, however, have focused on navigation for a 2D projected footprint of the robot moving in a projected 2D representation of the world. Therefore, they are unsuited for our motivating problem of mobile manipulation in a cluttered environment since they will reject any configuration of the robot where the 2D projected footprint of the robot is in collision with the 2D environment representation.

Hornung and Bennewitz [6] recently proposed an approach for efficient humanoid robot navigation by combining coarse 2D path planning in open spaces and detailed footstep planning in the vicinity of obstacles. This technique allows for finding solutions where planning based on a 2D grid fails.

Lazy collision checks and an enlarged robot model were used for fast RRT-based motion planning in a simulated kitchen environment by Vahrenkamp *et al.* [7]. The authors concentrated on manipulation tasks in a small part of the environment. They later extended the approach to coarse motion planning for the base while further away with an increased planner granularity closer to the goal [8]. However, their approach does not cover long navigation plans through a cluttered environment. Also, randomized planning approaches generally generate non-optimal solutions and rely heavily on smoothing to shorten the final path.

Marder-Eppstein *et al.* [1] use a compact 3D representation of the environment for a long running navigation demonstration. This representation allowed for quick updates for dynamic obstacles but still restricted the system to operate only in a projected 2D environment representation. Cart pushing with a mobile manipulation system was demonstrated by Scholz *et al.* [9] but navigation was again limited to goals where the 2D projection of the cart and the robot was not in collision. This disallowed, for example, actions that could move and store the cart under a table.

[1]As part of the Robot Operating System (ROS) at `http://www.ros.org/wiki/3d_navigation`

For efficient collision checking in 3D, several hierarchical approaches have been proposed in the past such as oriented bounding box trees [10] or sphere tree hierarchies [11], [12]. They all rely on a spatial subdivision of the 3D space. In contrast to these approaches, our collision checks are first performed in a multi-layered 2D map and we employ full 3D collision checks only when required.

Lau *et al.* [13] recently introduced techniques for incremental updates of collision maps in a non-circular robot's configuration space. After convolving a map with the discretized shape of the robot in a pre-computation step, collision checks correspond to a single lookup. While the map can be efficiently updated for dynamic environments, any change in the robot's kinematic configuration (e.g. after picking up objects) requires a full recomputation.

## III. The PR2 Robot Platform

We implemented our system on the PR2, which is a mobile manipulation system equipped with an omnidirectional base, an articulated sensor head and two 7-DoF arms. The sensors used throughout this work are the Hokuyo UTM-30LX 2D laser range finder in the base for localization and the stereo sensor in the head. The stereo cameras are augmented by a texture projector to provide dense data [14]. Calibrated joint encoders provide accurate proprioceptive data about the robot's kinematic configuration. The stereo sensor serves as the main sensing input for detecting obstacles in the environment for our approach.

The PR2's dense stereo pair has a narrow field of view of 55°. To make that practical for navigation, we implemented an active sensing behavior where the robot's sensor head always points in the current driving direction (which may be to the side or even backwards).

The sensing pipeline for our approach builds on existing components developed for a tabletop manipulation application [15]. The dense depth measurements of the PR2's stereo head are first run through a *self filter*. Based on the known robot mesh model and proprioception, points on the robot's body or on objects that are being manipulated or carried are removed. Accurate time synchronization between stereo and proprioceptive data on the PR2 ensures an effective self filtering. A *shadow filter* then removes all errors inherent to stereo sensing: points that are occluded from one of the stereo pair's cameras and veiling points on sharp edges and corners of the joints. Finally, a RANSAC-based ground-plane detection labels the remaining points as either ground or not ground for insertion into the octree-based 3D occupancy map described in detail next.

## IV. Environment Representation

Mobile manipulation requires the ability to represent and process a fairly large environment efficiently and accurately. The representation must be compact and easy to incrementally update to account for dynamic obstacles and changing scenes.

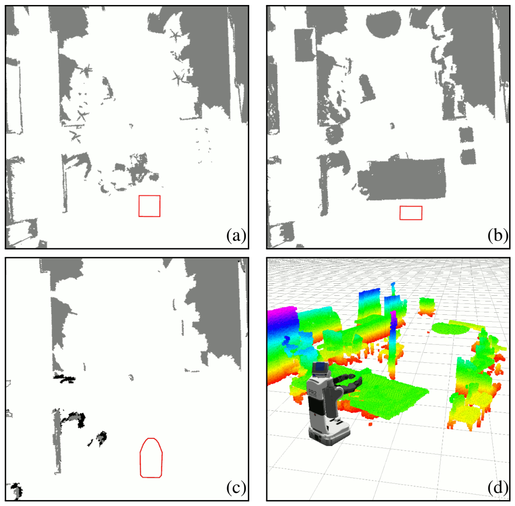

Fig. 2. The projected 2D grid map and footprint (red polygon) for each layer of the PR2 robot: base (a), spine (b), and arms (c). The full 3D occupancy grid from the octree representation is shown in (d).

## A. *Octree-based 3D Occupancy Map*

For collision checks between the robot and the environment, a full 3D representation is required. However, full 3D occupancy grids pose challenges on computational memory requirements if a high resolution is used for accurate results. Thus, we use the octree-based volumetric representation *OctoMap* [16] to efficiently build and store a probabilistic 3D occupancy grid at a resolution of 2.5 cm.

## B. *Continuous Map Updates*

For incrementally mapping the environment, the robot constantly inserts the most recent sensor information into the 3D map. The point clouds output from the perception pipeline are self-filtered as described in Sec. III, with points belonging to the ground plane marked. The map update is done using raycasting from the sensor origin. Knowledge about the ground points is hereby needed in order to clear the corresponding area and thus mark it as traversable.

We apply a probabilistic map update [16] to ensure the proper handling of sensor noise and dynamic updates when obstacles are appearing or disappearing (e.g., manipulated objects, or people moving in the environment).

The robot starts by building an initial 3D map of the environment that serves as a base representation which can then be modified by the incremental updates based on the robot's sensor data as it is executing its task. The resultant 3D map is the input to the motion planning framework, which further processes the map to create a multi-layered representation of the environment for improved efficiency in motion planning. We will now describe this process in further detail.

## C. *Multi-Layered 2D Obstacle Maps*

The geometric structure of a mobile manipulation robot such as the PR2 can be exploited for efficient collision checking by a spatial subdivision into multiple 3D volumes. Such a subdivision is especially applicable when the robot is navigating while carrying an object since its geometric structure will stay mostly constant during such a task. As illustrated in Fig. 1, the PR2 robot can be subdivided in such a manner into three parts: base, spine with head, and arms with possibly attached objects. Additionally, we also represent the environment in a compact manner using a multi-layered 2D representation. Each layer of this representation contains projected representations of the obstacles that can collide with the body parts associated with that layer. Thus, we represent the environment for the PR2 in three layers, each associated with one of the body parts mentioned above.

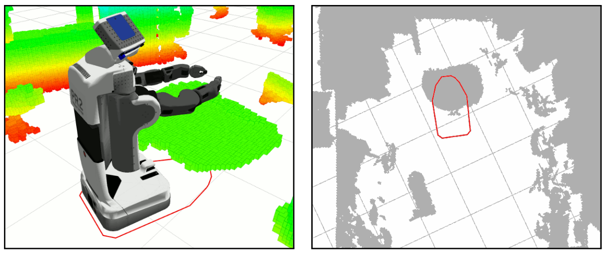

Fig. 3. *Left:* A snapshot of the PR2 with its arms over a table clearly indicates that the robot is not in collision with the table. *Right:* However, the footprint (red polygon) which is the convex hull of the 2D projection of the robot's configuration on the ground is in collision when using a single-layered grid map only.

Each incremental 3D map update also updates these down-projected 2D maps. The 2D projections are updated for a given map cell $(x, y)$ in layer $l_j$ by traversing all discrete levels $z_i$ in a range corresponding to the layer's minimum and and maximum height. For instance, for the layer used for the robot's arms, the height range is determined by the highest and lowest points on the arms and potentially attached objects. If for all $z_i$ in $l_j$, $(x, y, z_i)$ is free in the 3D map, $(x, y)$ is marked as free. As soon as one $(x, y, z_i)$ is occupied, $(x, y)$ is considered to be occupied. Otherwise, $(x, y)$ is marked as unknown space. Occupied cells in a 2D map layer hence stand for possible obstacles in 3D. Accordingly, a free cell guarantees that there is no collision in the 3D space corresponding to that layer.

In addition to the projected 2D map, we also have a down-projected footprint of the robot corresponding to the layer. Every time a plan is requested, we first update the 2D footprint for each layer by down-projecting the convex hull of the meshes for the body parts belonging to this layer (see Fig. 2).

The PR2 robot can be decomposed into the following layers (see Fig. 1): The box-like base, the spine from base through the head, and the arms with attached objects to be carried or manipulated. While the base and spine layers remain fixed for this robot, the arm layer may change its height and footprint as the arms move. Figure 2 shows an example of the three layers as projected grid maps with footprints. This representation was generated from a real scenario with the PR2 robot as shown in Fig. 2(d). It is clear that the arm layer has very few obstacles compared to the spine layer since the arms are at a greater height than most of the obstacles in the environment (e.g., the chairs and the table). Using multiple 2D layers in this manner allows motion planners to avoid expensive 3D collision checking.

An example is illustrated in Fig. 3 where the robot has its arms over a table and the base under a table. When using one single 2D map with a 2D projected footprint, the down-projected table will collide with the down-projected arms and base. A full 3D collision check would be needed to confirm whether a collision is actually occurring. With our approach, however, the table is only projected on the spine layer (which it doesn't collide with) and the arm and base footprints are not compared against it. Since none of the footprints are in collision, an expensive full 3D check is not needed.

Hence, using multiple layers allows us to declare the robot configuration collision-free more often without using expensive 3D collision checking. But we may also be able to determine that the robot is definitely in a 3D collision without having to do an explicit 3D collision check. This is explained in the following.

Some layers have the property that for each $(x, y)$ cell in the corresponding footprint and each $z_i$ in the layer, $(x, y, z_i)$ is part of the robot. This applies, e.g., to the box-like base of the PR2 and the tall spine. In these cases, if a 2D projection of an obstacle is inside the robot's footprint at the appropriate layer, the robot is definitely in collision with the object in 3D, regardless of the $z$ coordinate and a 3D check is not neeeded.

However, for certain layers such as the arm layer, this property cannot be used because they are not box-like and often change their shape. In some cases, we can still eliminate 3D checks by introducing the concept of a *tall obstacle*. A map cell $(x, y)$ is marked as a tall obstacle if all $(x, y, z_i)$ cells within the layer are occupied. Now, when the 2D footprint for that layer is in collision with this cell, the 3D configuration will always be in collision and a full 3D collision check is not needed.

Note that we apply our multi-layered 2D obstacle map for ruling out full 3D collision checks, not to replace them completely. In order to preserve the full flexibility of an arbitrary robot configuration, it may be still necessary to test its kinematic configuration for 3D collisions. Full 3D collision checking, when necessary, is performed using a collision model representation in the ODE simulation package [17].

## V. Planning & Navigation Framework

The 3D map and the multi-layered 2D obstacle maps serve as input to our planning and navigation framework. We apply a *global planner* to construct a plan in the position and planar orientation space $(x, y, \theta)$ to reach the goal. This plan is then executed by a *local planner*. To localize the robot during navigation, we currently rely on 2D Monte Carlo localization based on laser data. This localization uses a static map, which contains walls and other static parts of the environment, combined with odometry information from the wheels and an IMU located on the robot.

### A. Global Planning

*1) Search-based Planning on Lattice Graphs:* The global planner operates on a lattice graph [18], [19] corresponding

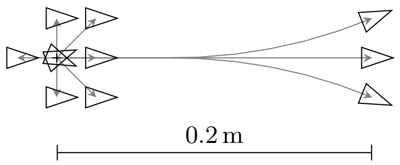

Fig. 4. Omni-directional motion primitives for the PR2.

to the 2D space with orientations $(x, y, \theta)$. Each state is connected to neighbors resulting from applying a set of motions primitives. Motion primitives are short, kinematically feasible motion sequences and the path found by the planner is a concatenation of these motions. The advantage of this state space representation is that the resulting plans tend to be smooth paths that can handle non-holonomic constraints and non-circular robot footprints. Fig. 4 shows the set of motion primitives we use for the PR2, including sideways and backwards motions in addition to driving forward and turning.

For efficient 2D collision checking, we compute a sequence of footprint collision cells for each motion primitive by rolling out the projected robot footprint along the primitive's path. This step took approximately 3 seconds in our trials and is only necessary if the robot configuration changes. For known configurations, it can be precomputed.

On the lattice graph, we then employ the Anytime Repairing A* (ARA*) search [20]. ARA* runs a series of weighted A* searches that inflate the heuristic component by $\varepsilon \geq 1$ while efficiently reusing previous information. The search starts out with a large $\varepsilon$, causing it to find a non-optimal initial solution quickly. Then, as time allows, the search reruns with incrementally lower values for $\varepsilon$ while reusing much of the information from previous iterations. Given enough time, ARA* finally searches with $\varepsilon = 1$, producing an optimal path. If the planner runs out of time before, the cost of the best solution found is guaranteed to be no worse than $\varepsilon$ times the optimal solution cost. This allows ARA* to find some solution faster than regular A*, and approach optimality as time allows.

ARA* checks the states generated at every step in the planning process for collisions using the multi-layered 2D map representation presented above. A 3D collision check is performed only when a possible collision is indicated in 2D. During the ARA* search, the costs for applying a motion primitive correspond to the length of the trajectory and additionally depend on the proximity to obstacles.

The heuristic for the planner uses a 2D Dijkstra search from the goal state. This heuristic only searches over the 2D grid map of the base layer with obstacles inflated by the base inner circle. Since we only search the base layer where 2D collisions imply 3D collisions, the heuristic stays admissible and consistent.

### B. Plan Execution

During execution, the concatenated discrete motion primitives from the global planner have to be converted into motor commands for the robot's base. Additionally, the validity

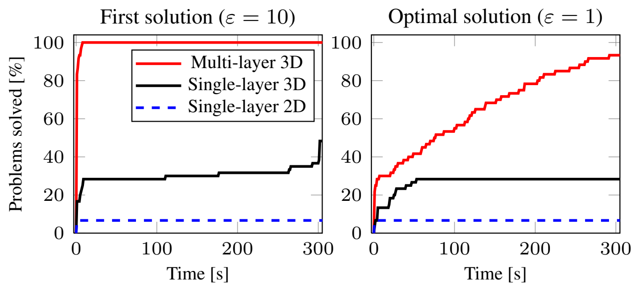

Fig. 5. Success rate for 60 planning problems in cluttered space. The percentage of solved problems after a certain time is shown for the first solution (*left*) and for the final, optimal solution (*right*).

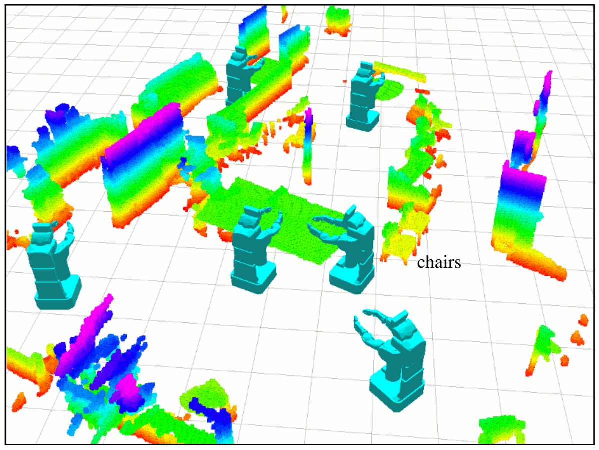

Fig. 6. The planning environment with six robot configurations chosen as start and goal poses. A narrow passage is formed by a table and two chairs, which were removed to create a second, easier scenario.

of the plan has to be checked while it is being executed because obstacle positions might change. To do so, the local planner computes the omnidirectional velocities required to reach the next $(x, y, \theta)$ pose along the path and performs a single trajectory rollout which is checked for collisions in the updated obstacle map. As in the global planner, collision checks are first performed in 2D, and only in 3D when required.

In case collisions are predicted, the local planner first tries to scale down the velocity command to reach the target pose as close as possible. For example, when docking with a table, a full discrete forward motion might collide with the table while a shorter, slow approaching motion reaches the table more closely.

In case the environment has changed or the robot has drifted from its path, the robot stops and the global planner is invoked again to find a new path around the obstacle.

## VI. Experiments

We carried out extensive experiments to evaluate our approach and demonstrate its usability for realtime mobile manipulation tasks.

### A. *Offline Motion Planning Tests*

The first set of experiments was designed to show the superior performance of our approach compared to conventional techniques. We generated multiple planning problems in which the robot was posed in a configuration with extended arms for manipulation. This configuration requires to use a large 2D projected footprint for traditional motion planning.

We evaluated the following three different approaches:

1) **Single-layer 2D:** A traditional navigation approach using only one 2D projected footprint of the robot and a 2D projected environment map.
2) **Single-layer 3D:** A navigation approach where 3D collision checks are always performed when the (single) 2D footprint is in collision with the 2D projected map.
3) **Multi-layer 3D:** Our approach using a multi-layered 2D occupancy grid and performing 3D collision checks only when absolutely necessary.

We performed experiments in two different scenarios. The first scenario is shown in Fig. 6. It includes a narrow passageway between two chairs and a table that the robot could only negotiate by moving sideways with its base under the table. For the second scenario the chairs were removed to make the planning problem easier. Note that the environment is a real cluttered office space and all the sensor data used for generating the environment representation came from a real PR2 robot.

Six different states (Fig. 6) were chosen manually and planning was carried out between each combination of start and goal from these states. This results in a total of 60 different planning problems across both scenarios. Planning was started with $\varepsilon = 10$ and allowed to continue until either $\varepsilon = 1$ or 5 minutes had passed. The success rate for each planning approach is shown in Fig. 5. Note that only two of the configurations in Fig. 6 have 2D footprints that are not in collision, enabling the traditional Single-layer 2D approach to succeed in only four cases.

Our approach (Multi-layer 3D) always succeeded in generating a motion plan, even in the most cluttered scenarios. It was able to perform far better than the Single-layer 3D approach and found far more solutions both in a short realtime window and in the long run (see Fig 5). For our approach, a first solution was always found within at most 7.3 s. Table I compares our approach and Single-layer 3D in the 32 planning attempts where both succeeded. Data from Single-layer 2D is omitted since it is only able to plan between two configurations. The results make it clear that using a multi-layered representation seriously improves the performance of the motion planner allowing it to find an initial solution very quickly. The refinement of the solution to a fully optimal solution ($\varepsilon = 1.0$) takes much more time but in practice the sub-optimal solutions already lead to an efficient navigation behavior of the robot. Table II shows data aggregated from the much harder 28 planning attempts for which only our approach (Multi-layer 3D) succeeded.

In Table I and II, the number of 2D and 3D collision checks refers to the number of motion primitives that had to be checked. A motion primitive may have several intermediate positions in addition to the start and end state. Each of our primitives may require up to 10 collision checks for individual robot positions. The actual number of collision

TABLE I

MEAN AND STANDARD DEVIATION FOR THE 32 PLANNING ATTEMPTS ON WHICH BOTH 3D APPROACHES SUCCEEDED.

| | Single-layer 3D | Multi-layer 3D |
|---|---|---|
| Time to first solution [s] | 130.81 | 0.03 |
| Standard deviation | 143.67 | 0.06 |
| Expands to first solution | 13 910.91 | 1 400.75 |
| Standard deviation | 16 572.31 | 1 618.08 |
| $\varepsilon$ after 5 s | 1.5 | 1.00 |
| 2D collision checks (primitives) | 153 375.06 | 15 408.25 |
| 3D collision checks (primitives) | 22 017.09 | 0.00 |

TABLE II

MEAN AND STANDARD DEVIATION FOR OUR APPROACH IN THE 28 PLANNING ATTEMPTS ON WHICH ONLY MULTI-LAYER 3D SUCCEEDED.

| | |
|---|---|
| Time to first solution [s] | 1.52 |
| Standard deviation | 2.13 |
| Expands to first solution | 69 915.39 |
| Standard deviation | 107 906.74 |
| $\varepsilon$ after 5 s | 6.28 |
| 2D collision checks (primitives) | 769 069.32 |
| 3D collision checks (primitives) | 2.89 |

checks for each motion primitive may be less since we abort as soon as one intermediate position is in collision.

The number of motion primitives that checked for 3D collisions with our approach is very small because having multiple layers allows the inflation (which penalizes the cost function for the robot as it gets closer to obstacles) to be more informative in keeping the search away from obstacles and 3D collision checks. Additionally, as can be seen from Fig. 2, there are very few obstacles in the arm layer of the robot that are not classified as tall obstacles. However, as the planner approaches $\varepsilon = 1.0$, even our approach eventually has to do more 3D collision checks. This is why, on average, we only reach $\varepsilon = 6.28$ in 5 s in some of the harder cases and in many of the cases 10 or 100 thousands of collision checks are required before we reach the optimal solution.

### B. Docking Maneuvers With Real Robot

A second set of experiments involved repeated docking and undocking with a table using the real PR2 robot in a similar cluttered office environment. This is the kind of action that a robot will have to execute when it needs to pickup or place objects on the middle of the table. Note that configurations that require the robot to reach the middle of the table are unachievable using traditional 2D navigation approaches. We evaluated this set of experiments solely for our approach to motion planning with a multi-layered representation.

A typical docking maneuver to pick up a large object from a table can be seen in Fig. 7. In our experiments, all goal configurations were reached collision-free while the robot docked and undocked a table 12 times.

### C. Navigation While Carrying Large Objects

Finally, we evaluate our approach on a set of experiments for a difficult scenario involving autonomous navigation within an extremely cluttered environment while carrying a

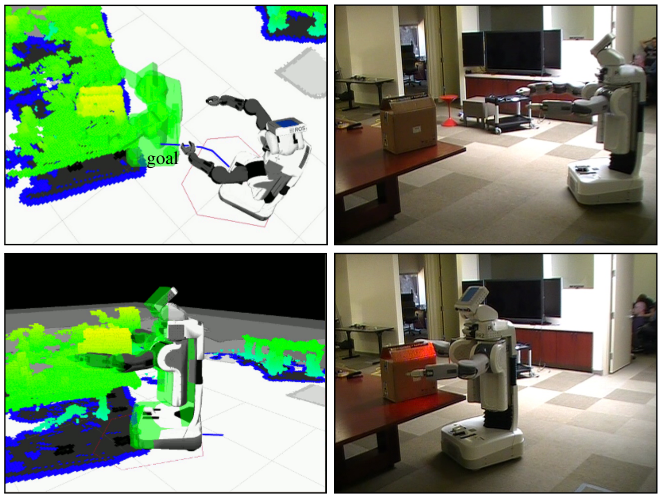

Fig. 7. Docking with a table to pick up a large object. *Left*: Goal configuration (shaded green) and resulting plans (blue). *Right*: Execution of the plan by the PR2.

laundry basket with both arms. In this scenario, the robot often had to pass through narrow areas, forcing it to move sideways close to obstacles. The location of the laundry basket was provided a priori to the robot. The robot first planned a path to a goal in front of the basket. It then lifted up the basket with a pre-defined motion of the arms. A bounding cylinder was used to filter out the basket from the robot's sensor data. The final navigation goal for the robot was then near another table where it would have to put the basket down again. We conducted multiple trials using several combinations from a set of six goal and start states. The robot successfully executed each task.

Fig. 8 shows a series of snapshots while executing this task, whereas the complete planned path is shown in Fig. 9. The precomputation time for the footprints was 3.46 s in this scenario and the complete planning time 3.16 s. We planned to the first solution in this case ($\varepsilon = 10$). To pick up the laundry basket, the robot initially had to move its base under the table. Afterwards, the narrow passage with chairs forced it to move sideways under the table with the basket over the table. Finally, the PR2 approached the second table to put the basket down. Note that large parts of the workspace in this scenario are unreachable by the robot using traditional 2D navigation planning approaches.

## VII. CONCLUSIONS AND FUTURE WORK

We presented a novel approach for efficient navigation with a mobile manipulation robot in three-dimensional, cluttered environments such as offices or homes. Our integrated approach combines an efficient octree-based 3D representation and an anytime search-based motion planner. For efficient collision checking during planning, our approach utilizes a multi-layered 2D environment representation, allowing the generation of almost real time bounded sub-optimal plans.

We demonstrated the performance of our framework on the PR2 mobile manipulation robot in exhaustive real-world experiments. The robot was able to navigate efficiently and

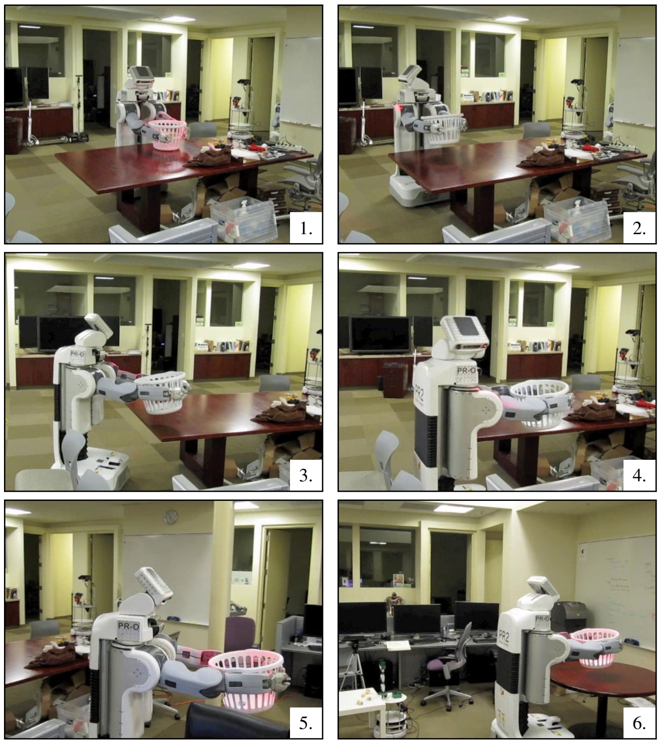

Fig. 8. Sequence of snapshots showing the PR2 robot navigating autonomously with a laundry basket in a cluttered environment. A video of this sequence is available online.

collision-free, carrying a laundry basket with both arms from one table through a heavily cluttered room to another table. The robot also successfully performed docking and undocking maneuvers with extended arms at a table, thus maximizing its workspace for manipulation by moving the base under the table. Our approach found an initial solution on average within 2 seconds while the highest planning time for an initial solution was about 7 seconds. This underlines the efficiency and practicability of our approach for navigation in mobile manipulation scenarios.

A natural extension for future work will be to move larger objects such as chairs or carts with articulated primitives [9] in a cluttered environment, e.g. to store them away under a table.

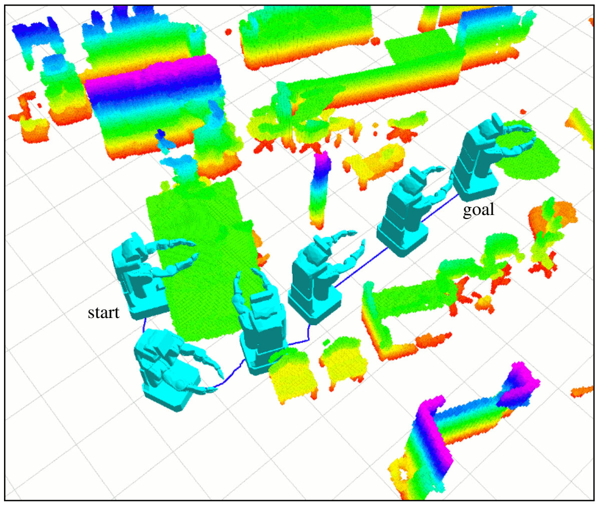

Fig. 9. Visualization of the 3D environment and path taken by the PR2 robot in moving a laundry basket through a cluttered indoor environment. (Note: The basket itself is not visualized.)

## REFERENCES

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

[2] M. Likhachev, http://www.ros.org/wiki/sbpl, 2010.

[3] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[4] I. R. Nourbakhsh, C. Kunz, and T. Willeke, "The mobot museum robot installations: A five year experiment," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[5] R. Kümmerle, D. Hähnel, D. Dolgov, S. Thrun, and W. Burgard, "Autonomous driving in a multi-level parking structure," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

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

[7] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Efficient motion planning for humanoid robots using lazy collision checking and enlarged robot models," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[8] N. Vahrenkamp, C. Scheurer, T. Asfour, J. J. Kuffner, and R. Dillmann, "Adaptive motion planning for humanoid robots," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

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

[10] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," in *Proc. of ACM SIGGRAPH*, 1996.

[11] K. Steinbach, J. Kuffner, T. Asfour, and R. Dillmann, "Collision and self-collision detection for humanoids based on sphere tree hierarchies." in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2006.

[12] S. Quinlan, "Efficient distance computation between non-convex objects," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1994.

[13] B. Lau, C. Sprunk, and W. Burgard, "Incremental updates of configuration space representations for non-circular mobile robots with 2d, 2.5d, or 3d obstacle models." in *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2011.

[14] K. Konolige, "Projected texture stereo," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[15] 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 *ISER*, New Delhi, India, December 2010.

[16] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010, software available at http://octomap.sf.net/.

[17] "Open dynamics engine," http://www.ode.org.

[18] M. Likhachev and D. Ferguson, "Planning long dynamically-feasible maneuvers for autonomous vehicles," in *Int. Journal of Robotics Research (IJRR)*, 2009.

[19] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[20] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* search with provable bounds on sub-optimality," in *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2003.