Towards Navigation in Three-Dimensional Cluttered Environments

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

I. Introduction

Home assistance is one 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 frameworks already enable fast planning and robust execution for a 3-DoF robot base in a 2D grid map indoors [1]. Sensor data is either two-dimensional or projected down to 2D from 3D. However, this usually means that motion plans cannot be generated to goal locations in highly cluttered areas or to goal locations where the 2D projection of the robot seems to be in collision. The robot thus loses capabilities such as approaching a table for a manipulation task, which requires partly moving its base under the table, and can not navigate with extended manipulators, e.g. to carry a tray, without keeping an even larger distance to obstacles due to the enlarged footprint.

To overcome these limitations, we propose a three-dimensional navigation framework which performs full 3D collision checks for arbitrary joint configurations of a mobile manipulation robot with a 3-DoF base. For efficiency, we rely on a down-projected 2D map and perform expensive 3D collision checks only when needed. The robot incrementally builds a 3D occupancy map in an efficient octree-based representation. A global path planner yields an anytime solution on a lattice graph. The local planner then executes this path and validates it during execution. Near three-dimensional obstacles, a full 3D collision check between the robot model and the obstacle map is performed.

We validate our approach through experiments on the PR2 mobile manipulation robot. The PR2 robot has an omnidirectional base and two arms. Using our approach, we were able to perform docking and undocking maneuvers with extended arms at a table (Fig. 1). Our framework thus allowed the PR2 to reach a far bigger workspace by allowing the base to move under overhanging obstacles and also into cluttered areas. Our framework builds upon and extends the capabilities of the Search-based Planning Library (SBPL, [2]) and is available as open source software.

II. Environment Representation

For collision checks between the complete robot model and the environment, a 3D representation is required. However, full 3D occupancy grids pose challenges on the memory requirements if a high resolution for accurate results is used. Thus, we use the octree-based volumetric representation of OctoMap [3] to efficiently build and store a probabilistic 3D occupancy grid which also accounts for unknown space. For incremental mapping, the robot constantly inserts its

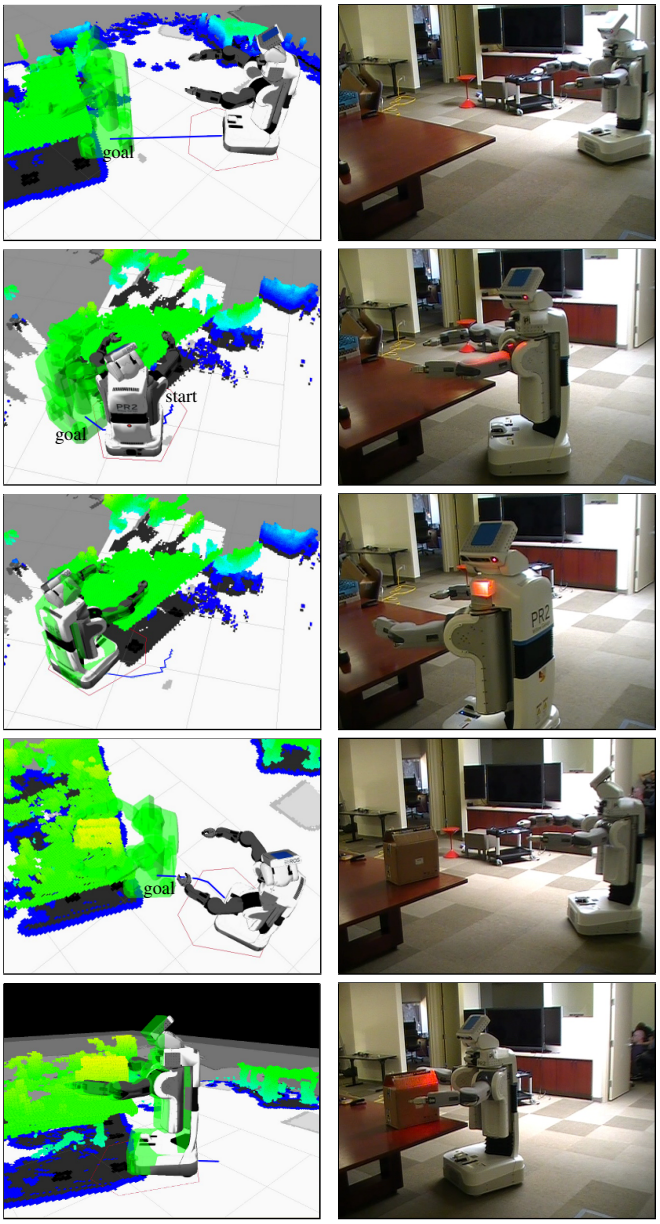

Fig. 1. The PR2 docks and undocks a table with untucked arms, thereby considering its full body pose for avoiding three-dimensional obstacles observed by its sensors. Left: Goal configuration (shaded green) and resulting plans (blue). Right: Execution of the plan by the PR2.

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

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 and by Willow Garage Inc.

3D sensor information as point clouds into the map, which is then updated using raycasting on the individual beams. A probabilistic map update ensures 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 3D occupancy map is then used for collision checks. Each map update also updates a down-projected 2D map of the environment that is used for efficiently computing collision-free motions.

III. Planning & Navigation Framework

Besides incrementally building and updating the map, our framework for 3D navigation consists of a global planner to construct a globally optimal plan and a local planner to execute and validate this plan.

A. Global Planning

1) Search-based Planning on Lattice Graphs: Our global planner operates on a lattice graph [4] corresponding to the 2D space with orientations (x, y, θ). This avoids the typical problems of dense grid discretizations by building a graph of concatenated motion primitives of the robot platform, each consisting of a feasible, possibly omnidirectional motion and resulting in a (x, y, θ) pose. On the PR2 robot, this means that the robot can explicitly plan for sideways and backwards maneuvers with or without changing its orientation.

On the lattice graph, we then employ the Anytime Repairing A* (ARA*) search [5]. ARA* runs a series of weighted A* searches while efficiently reusing previous information. Each weighted A* trades off optimality for speed with an ε-suboptimal heuristic, and can be orders of magnitude faster than A*. ARA* iteratively lowers the ε until either the optimal result at $\varepsilon = 1$ is found or the given planning time is over, resulting in an efficient anytime-capable A* search.

2) 3D Collision Checks for Efficient Planning: Collision checks are one main factor influencing the planning time, as they occur at every search node expansion. Thus, they need to be as efficient as possible. For efficient planning, we combine the 3D map with the down-projected 2D map and check collisions in 3D only when needed. Whenever the planner expands a node and determines the costs of the expansion, a 2D collision check of the robot's down-projected footprint (including all possibly extended links) is done in the 2D map first. If no collision is reported, then it is guaranteed that there is also no 3D collision. Otherwise, a 3D collision check between the full robot configuration and the 3D occupancy map is initiated and determines the actual validity of the expanded motion. For this collision check between the robot's links (given as 3D model meshes) and the occupied voxels of the 3D map, we currently use ODE.

This approach to collision checking ensures that expensive 3D collision checks need to only be carried out for capabilities like moving through clutter or moving close to tables. When leaving them away, e.g. due to planning time constraints, the planning framework gracefully degrades to 2D planning with footprint collision checks.

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. Also, the validity of the plan has to be checked while it is being executed because new or previously unseen obstacles might appear. To do so, the local planner computes the omnidirectional velocities required to reach the next (x, y, θ) pose along the path and does a single trajectory rollout which is checked for collisions in the updated collision map. Just as in the global planner, the collision check is first performed in 2D, and only performed in 3D when required.

In case collisions are predicted, the robot stops and the global planner is invoked again in order to find a new collision-free path.

IV. Results

For evaluation, we employed a PR2 robot in a cluttered indoor laboratory environment. We used the dense stereo sensor (augmented by a texture projector) of the PR2 for mapping at a resolution of 2.5 cm. During navigation, the PR2 left its arms unfolded. This is a useful pose e.g. for carrying trays, pushing movable objects such as carts, pick and place, or other mobile manipulation tasks. Within our framework, the actual body configuration is arbitrary and can be changed at any time.

Some typical results can be seen in Fig. 1. All goal configurations were reached collision-free. For paths in open spaces and when docking or undocking the table, ARA* yielded the optimal planning result with a final $\varepsilon = 1$. Only for complicated scenarios, such as starting and ending nearly in collision with the table or a box on the table (Fig. 1, last four rows), a non optimal path was returned within the maximum planning time of 15 s. In these scenarios, nearly all node expansions trigger expensive 3D collision checks. However, the path was still reasonably short and sufficiently smooth due to the local planner. In our current work, we are working on a plan refinement during execution to yield even better results in complex scenarios.

All source code is available in the Robot Operating System (ROS) at `www.ros.org/wiki/3d_navigation`.

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] 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/.

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

[5] 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.