Mobile Manipulation in Cluttered Environments with Humanoids: Integrated Perception, Task Planning, and Action Execution

Armin Hornung Sebastian Böttcher Jonas Schlagenhauf
Christian Dornhege Andreas Hertle Maren Bennewitz

***Abstract*— To autonomously carry out complex mobile manipulation tasks, a robot control system has to integrate several components for perception, world modeling, action planning and replanning, navigation, and manipulation. In this paper, we present a modular framework that is based on the Temporal Fast Downward Planner and supports external modules to control the robot. This allows to tightly integrate individual sub-systems with the high-level symbolic planner and enables a humanoid robot to solve challenging mobile manipulation tasks. In the work presented here, we address mobile manipulation with humanoids in cluttered environments, particularly the task of collecting objects and delivering them to designated places in a home-like environment while clearing obstacles out of the way. We implemented our system for a Nao humanoid tidying up a room, i.e., the robot has to collect items scattered on the floor, move obstacles out of its way, and deliver the objects to designated target locations. Despite the limited sensing and motion capabilities of the low-cost platform, the experiments show that our approach results in reliable task execution by applying monitoring actions to verify object and robot states.**

I. Introduction

Robotic assistants should be able to autonomously solve and execute their tasks. We see this trend also in the current focus of the DARPA Robotics Challenge, to encourage for more autonomy instead of pure tele-operation as in the pre-final trials. To enable full autonomy, a robot needs to learn and maintain a model of all relevant aspects of the environment, to reason about its goals and its own actions as well as their effects. Furthermore, it must robustly execute the planned actions in the real world.

In this paper, we address autonomous mobile manipulation with humanoids in cluttered environments. In particular, we consider the task of collecting objects and delivering them to designated places in a home-like environment cluttered with obstacles. We present a modular system that tightly integrates perception, world modeling, navigation, manipulation, and action planning and replanning with a high-level symbolic planner to solve this task.

In the application used to evaluate our approach, the robot has to collect toys that are scattered on the floor, move obstacles out of its way, and navigate to target locations, as illustrated in Fig. 1. Objects that need to be tidied up are picked up and carried by the robot. Other small objects blocking the way are considered as clutter and will be locally moved aside. Our approach is implemented on a real humanoid that autonomously acts in the environment. The robot uses onboard sensing to perceive the environment and to continually monitor its action execution.

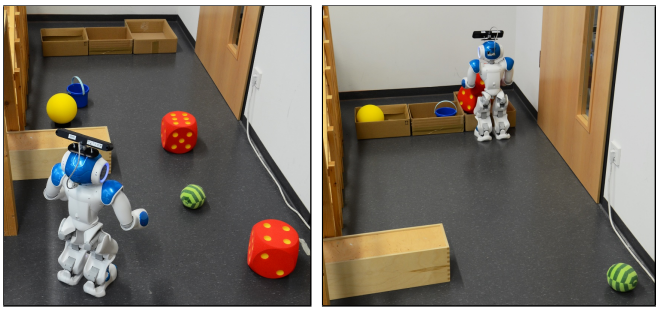


Fig. 1. Left: Example tidy-up task to be solved by our integrated system. All larger objects have to be picked up and placed into their corresponding container. To complete this task, the robot has to reason about a sequence of actions such as picking up an object, walking to a target location, and moving an object out of the way. Right: The final state with all objects tidied up, except for the green ball that was considered as clutter and moved aside by the robot with its foot.

Note that simpler pick and place scenarios in open spaces can potentially be solved with a scripted system or using a state machine that follows the greedy heuristic of always picking up the closest object and cleaning it up. In this paper, however, we focus on a more sophisticated approach that plans actions on a symbolic level and also carries out observation actions to verify the state of the world during manipulation. More complex planning is needed, for example, in cluttered environments where spatial constraints between nearby objects need to be considered. Such scenarios require the robot to plan the complete task on a *symbolic* level, e.g., to move aside a blocking object and continue cleaning it up at a later stage. To this end, we build upon and extend our TFD/M planning framework [1] towards mobile manipulation with humanoids. Since the robot has to deal with uncertainty about the current world state and failures in action execution, our system replans the actions if necessary as detected by execution monitoring to verify object and robot states.

In experiments with a Nao humanoid, we demonstrate the capabilities of our planning approach and its reliability despite the motion and sensing limitations of this low-cost platform. Using our approach, the robot is able to tidy up a cluttered environment while avoiding obstacles and moving blocking objects out of its way. In our framework,

All authors are with the Dept. of Computer Science, Univ. of Freiburg, Germany. M. Bennewitz is also with the Humanoid Robots Lab, Univ. of Bonn. This work has been supported by the German Research Foundation (DFG) within the SFB/TR-8 *Spatial Cognition*, the PACMAN project within the HYBRIS research group (NE 623/13-1), and the *BrainLinks-BrainTools* Cluster of Excellence (grant number EXC 1086) as well as by the European Commission under contract number FP7-610532-SQUIRREL.

we integrate perception, world modeling, action planning and replanning, navigation, and mobile manipulation of multiple objects to realize truly autonomous behavior for humanoid robots.

II. Related Work

Stilman and Kuffner introduced the concept of *navigation amongst movable objects* (NAMO) [2]. Their method allows a robot to manipulate its surroundings in order to reach the goal. The NAMO approach decomposes the configuration space and the underlying structure of the navigation problem into disjunct regions of accessibility. The planner finally searches for individual obstacle motions that connect two disjoint regions and allow the robot the transition from one region to the other. To provide fast results suitable for navigation, the NAMO planner utilizes the connectivity structure of these regions in a reduced problem domain. The specialized domain of the planner prevents an application to solve general tasks, e.g., cleaning up all objects. Stilman *et al.* later employed NAMO on the humanoid robot HRP-2 to plan and execute a path through an environment with movable chairs and tables [3]. The world state here was completely observable from an external motion capture system. NAMO was furthermore extended towards the exploration of unknown space [4] and decision-theoretic planning under uncertainty by means of hierarchical reinforcement learning [5].

Levihn *et al.* recently introduced *environment aware planning*, in which the robot may use parts of the environment to reach navigation goals that are otherwise not reachable [6]. In this approach, a path planner is allowed to violate certain constraints, such as a too high step height for climbing. The violation is then locally resolved by modifying the environment at the corresponding location, e.g., by placing a small step that can be climbed in front. Similar to NAMO, this heuristic is suitable for a reduced problem domain where a violation can be resolved by moving a single object.

More general and domain-independent approaches combine task planning at a symbolic level with geometric reasoning. Wolfe *et al.* considered pick-and-place tasks where task and kinematic planning are combined [7]. The authors proposed a hierarchical task network to model and take into account lower-level synergies and conflicts. Kaelbling and Lozano-Pérez plan for mobile manipulation in the belief space to account for uncertainty about the current world state and about the outcome of actions [8]. The authors use a joint Gaussian distribution to represent the object and robot poses and explicitly consider perception actions to reduce uncertainty. Their approach is able to generate intermediate poses for objects during planning similar to ours. Gaschler *et al.* also consider geometric entities during planning by modeling the world using 3d volumes [9].

Dornhege *et al.* discussed integrated task and motion planning for the PR2 robot [10], [11], [12]. Their approach combines classical symbolic planning with geometric reasoning in the TFD/M planner with semantic attachments [1]. To deal with uncertainty about the current world state and failures in action execution, Dornhege *et al.* apply execution monitoring

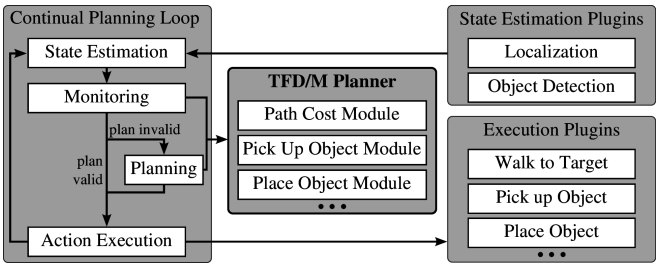


Fig. 2. Overview of the TFD/M planner in the continual planning loop with its modules and plugins.

of actions and replanning if necessary. In the work presented in this paper, we use the TFD/M planning framework and extend it for pick-and-place tasks with humanoid robots. The focus here is on navigation for mobile manipulation in a domain where various objects have to be tidied up and where the robot has to find its way through clutter.

III. Task Planning

At the core of our system lies the symbolic planner *Temporal Fast Downward/Modules (TFD/M) with semantic attachments* [1]. The planner with the modules and plugins used in this work is illustrated in Fig. 2. The planning task, in our case cleaning up all known objects, is specified in the PDDL format [13]. Planner plugins for state estimation continuously update the robot state from the localization estimate and the state of objects with a combined RGB and depth perception. Modules are external procedures called by the planner for reasoning at the geometric level, e.g., to determine navigation costs between different locations in the environment. Once a symbolic plan is found, the plan is executed with the execution plugins while being continuously monitored by the planner. Whenever the current plan is no longer valid, e.g., because the robot dropped a carried object or new objects that influence successful task execution are detected, the TFD/M planner replans.

A. Planning Using TFD/M

As described above, we employ *continual* planning, where the task planner is embedded in a state estimation, monitoring, planning, and execution loop to be able to react to unforeseen events in a real-world setting. After each executed action, our approach estimates the current world state and checks whether the remaining actions of the plan can still be executed, i.e., their preconditions are fulfilled. If this is not the case, replanning is triggered to adapt to the new situation. TFD/M is an anytime planner, i.e., it computes a sub-optimal plan quickly and improves it until a timeout is reached or the optimal plan is found.

In addition to the capabilities of a classical planning system, our TFD/M planner features so-called semantic attachments [1]. To capture the geometric aspects of robotic tasks, the semantic attachments specify conditions and effects that to the planner appear to be symbolic, but are implemented by an external procedure. For example, the cost value of a *walk* action is determined by calling the path planner from Section IV, which returns infinite costs in case the

path between start and goal is blocked (for efficient reuse, such computations are cached [11]). In this way, plans are guaranteed to be sound on the symbolic *and* geometric level. Additionally, our planner generates different alternatives for certain actions during planning, e.g., an intermediate location to put down an object does not need to be specified in advance (see Sec. VII).

B. Planning Domain

The planning domain defines the symbolic actions available to the planner. Each action specifies its cost, under what conditions it can be applied, and what its effects are. In our application, we use the following actions:

- The action *detect_objects* is initially executed at the robot's starting pose to fill the planner with potential goals. The system can also plan robot actions to reach pre-defined scan locations from which *detect_objects* is executed to provide a better viewpoint on the environment. In addition, the action is executed after a monitoring failure (e.g., after losing a carried object) to account for possible changes in the object positions. After execution, the robot updates the poses of known objects and adds newly discovered ones to the planning task. Whenever this action changes the world state, e.g., by adding a new object, replanning is triggered.
- The action *walk_to_target* can be executed given there exists a collision-free path as computed by the navigation planner (see Sec. IV).
- To acquire objects, the action *pick_up_object* is performed with one or two hands, depending on the object type. This action requires the corresponding hand(s) to be empty and the robot to be close to the manipulated object.
- When holding an object, the robot can place it into a container or at an intermediate location (determined according to Section VII) using the action *place_object*.
- Small obstacles blocking the robot's path ("clutter") can be moved out of the way with the feet if the robot is close enough using the *move_obstacle* action.

The goal for the tidy-up task is defined as the state in which each object to be tidied up is put in its corresponding container. The costs of each action correspond to the estimated time of executing the action and were experimentally determined.

IV. Autonomous Navigation

To estimate the 3D pose of the robot in the environment, we employ a particle filter using kinematic odometry of the robot's walking engine as well as data from the onboard depth camera and IMU sensor [14]. We here use a 3D map of static obstacles, which is known beforehand. Based on the estimated pose, the robot then continuously integrates 3D range data into a second representation that contains all other detected objects in addition to the static obstacles [15]. Our system uses this continuously updated world representation for obstacle avoidance and path planning.

For navigation planning, we employ the ARA* planner [16] to plan a 2D path in a 2D projection of the dynamically updated 3D map, with the vertical range of the 2D projection corresponding to the robot's operation height. In addition to path planning for reaching the next target location during execution, the task planner uses ARA* in the path cost module to obtain the costs between two locations in the world state. The path costs are hereby scaled corresponding to the execution time of the robot when following the 2D path.

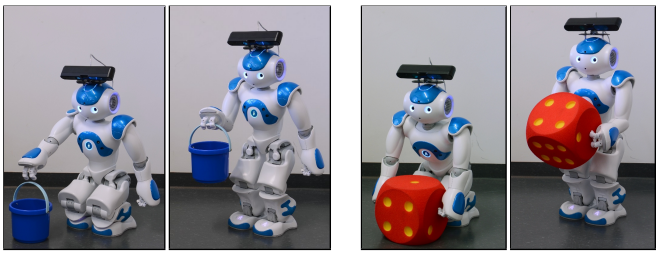


Fig. 3. Whole-body motion for picking up a bucket by grasping its handle (left) and for picking up a cube with both arms (right).

V. Object Detection

To detect objects, the robot uses RGB-D data acquired with its onboard sensors. In the point cloud, our approach first performs a ground plane segmentation and then clusters the remaining points into object hypotheses. To detect and classify the actual object belonging to each cluster, we employ a Haar features-based cascade classifier on the RGB image. From the co-registered depth image, we then obtain the actual location of the object in the 3D point cloud. Note that a more general object detection method can easily replace our current technique.

By using the robot's pose estimate and the kinematic estimate of its joint configuration, we finally obtain the global pose of each detected object, which is then added to the planner world state. Any remaining cluster that was not classified as one of the objects to clean up is either a static obstacle or clutter if it is sufficiently small.

VI. Manipulation and Grasp Verification

Currently, we rely on a set of pre-defined whole-body motion primitives to manipulate objects (see Fig. 3 for examples). These primitives consist of whole-body joint trajectories corresponding to each object type, which we demonstrated to the robot with kinesthetic teaching. While the motions are executed open loop and thus need accurate positioning in front of the object, they require no expensive whole-body planning and result in reliably picking up objects while keeping balance.

We assign four possible manipulation locations to objects regarded as clutter, from which the robot can locally move the object aside with its feet. Similarly, for each object to be cleaned up, we consider four possible pickup locations. Larger objects such as cubes and balls have to be grasped with both arms, while a bucket for example can be grasped at its handle with the left or right arm, depending on which location is better accessible with the path planner. To pick up an object, the robot first approaches the pickup location based on its localization estimate. Once it is sufficiently close, the

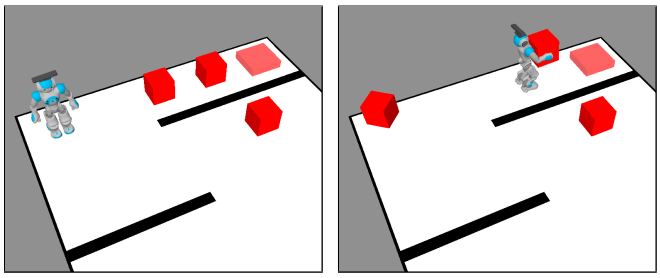


Fig. 4. A clean-up scenario in which the red cubes have to be transported into the container at the top right. The object configuration does not allow the robot to clean up one object after another (left). Instead, it has to move a blocking cube out of the way first (right). The symbolic planner samples and grounds intermediate locations such as the one in the corner as needed.

robot switches to a local servoing mode for a higher accuracy. Here, it relies on a close-range RGB perception to detect the object in its local coordinate frame and repositions until it reaches the object's pickup location.

After executing the corresponding grasping action, the robot actively verifies the success of the action. To this end, we employ an evaluation of the measured arm joint angles or a vision-based verification depending on the type of object. Occasionally, the robot may also lose a carried object while walking. When a loss of a carried object is detected, the robot stops the current navigation plan, walks back a few steps and performs an observation action to re-detect the object. Then, the world state of the planner is updated and our system replans to pick up the lost object again.

VII. Foresighted Object Placement

In some planning instances, objects may block the path to a pickup or delivery location, so that the robot cannot clean up the objects one by one. The robot then needs to temporarily move an object away without impeding future navigation plans. Unlike clutter that is just moved aside, graspable objects need to be cleaned up at a later stage. Such a scenario is depicted in Fig. 4. For efficiency, however, the planner cannot consider all possible locations in the environment for temporarily storing objects as this would exponentially increase the planning state. We instead sample object locations close to the robot that are not impeding paths through the environment. For this purpose, the robot learns a placement cost map for the environment beforehand by applying its path planner between sampled start and goal poses and counting how often each part of the environment is traversed by paths. Fig. 5 shows the object placement costs for an example environment. Areas close to outer walls and in corners receive low costs, while narrow passages and other parts that are regularly traversed receive high costs.

The symbolic planner regularly samples intermediate locations, hereby preferring low placement costs. A grounding process inserts a fixed number of these locations as so-called symbolic objects into the planning state if they are collision-free with respect to other objects in the environment. In each sampling iteration, we increase the maximum distance of sampled locations from the robot to ensure a probabilistic completeness of the planner.

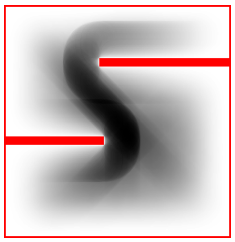


Fig. 5. Object placement costs from motion planning between sampled start and end points in the environment of Fig. 4 (darker: higher costs).

VIII. Experiments

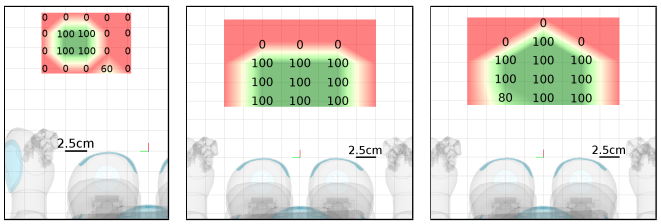
For the experiments presented in this paper, we use a Nao humanoid that is equipped with a head-mounted Asus Xtion Pro Live RGB-D camera for 3D perception. The camera is mounted so that its optical axis faces the floor in a 30° angle while walking. We found this the best compromise between observing the near range for obstacle detection and looking ahead for localization. Except for the acquisition of sensor data, we performed all further processing and planning offboard on a standard PC. For the experiments, we used three different object types, for which we trained Haar classifiers. The number of objects of a specific type in the environment is not known by the robot in advance.

We used the standard walking controller of the Nao to follow the planned path through the environment. To make the walking gait more robust and avoid a fall, we adjusted the gait configuration (COM height, step frequency, and step width) depending on which object the robot currently carries. In addition, the robot needs to tilt back its head with the camera to counter the weight of a bimanually carried object.

In our experiments, we use the anytime TFD/M planner to find the first solution, which is potentially sub-optimal with respect to the execution time, and continue searching for another 15% of the elapsed computation time (at least 10 s) to improve the solution.

A. Picking Up Objects

We first evaluated the success rate of picking up objects using the whole-body motion primitives. To this end, we placed three different objects at different positions relative to the robot. In our example scenario, we used a small bucket that has to be grasped at its handle, a soft ball, and a large cube made of foam. Fig. 6 shows the success rate of picking up the three objects with the motions illustrated in Fig. 3. Despite the open-loop motions, the robot was able to successfully pick up all objects within a margin of a few centimeters. In all cases, the verification of the grasp, either by monitoring the arm joint angles or by visually detecting the object in the hand as in the case of the bucket, returned the correct state of the object. Note that during execution, the high-level planner invokes relocalization and repositioning of the robot in case of a failed pickup attempt.



(a) Bucket (b) Ball (c) Cube

Fig. 6. Success rate for picking up a bucket, a ball, and a cube at different positions relative to the robot (N=5 tries for each location). The numbers indicate the success rate of the object placed at the corresponding location while the robot remains at the same location, as seen from the top.

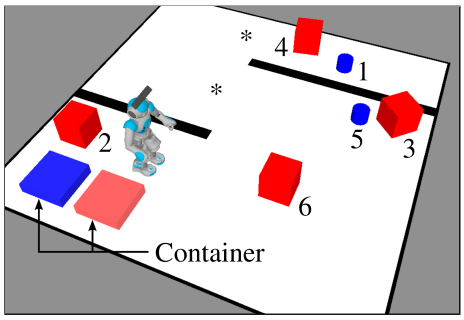


Fig. 7. Nao humanoid in a simulated environment. The clean up task is complete when the robot has placed all of the six objects into their corresponding container (Scenario 1). Optional movable obstacle locations are marked with a * and are used in Scenario 2.

B. Planning Time

Next, we performed simulation experiments to evaluate the planning performance. Here, the robot navigates and executes all actions open-loop and has perfect world knowledge. The simulation environment is depicted in Fig. 7 and has a size of 2.5 m $\times$ 2.5 m. In Scenario 1, the robot had to pick up six objects that are either cubes (red) or buckets (blue) and transport them to the corresponding container. Some of the objects prevented the action of picking up or putting down another one. For example, Bucket 5 blocked picking up Cube 3 and Cube 2 blocked access to the container for buckets (blue). The planner thus had to find a valid order of the actions to execute by employing the path planning module to check for accessibility. Note that, in general, each object can be picked up from four different locations next to it.

Table I shows the planning and execution times in the simulation environment for different numbers of objects (Scenario 1). A planning run with N objects hereby contains the objects numbered $1 - N$ from Fig. 7 in addition to the two containers. The planning time scales exponentially with the number of objects, since for every additional object the planner can rearrange the order in which objects are picked up and transported.

TABLE I

PLANNING AND EXECUTION TIME IN THE SIMULATION ENVIRONMENT

Scenario	Num. objects	Planning time [s]	Execution time [s]
Scenario 1	6	265	935.71
	5	155	783.20
	4	37	622.57
	3	18	421.23
	2	4	290.53
	1	1	214.02
Scenario 2	3	12	343.51
Scenario 3	3	71	456.52

In a second scenario in the same environment, the robot had to clean up only Cylinder 1 (all other objects were removed) but with two additional movable obstacles considered as clutter (marked by *) that blocked the path. Such clutter has to be moved aside to allow for traversal, which corresponds to a NAMO setting [3]. Our planning system was also able to successfully solve this task, with the planning and execution times shown in Table I as Scenario 2.

In Scenario 3 depicted in Fig. 4, three cubes had to be cleaned up into the container at the top right. However, the path to the container was blocked so that the robot could not clean up the objects one by one. Instead, it had to generate intermediate locations during planning using the foresighted object placement approach described in Sec. VII. The cost map from which new locations were generated is displayed in Fig. 5. Note that the planning time for this task was higher than for three objects in Scenario 1 because of the additional placements to be considered.

C. Integrated System

In a final experiment, we evaluated our approach in a real environment with a Nao humanoid. Fig. 8 shows snapshots from a video accompanying this paper. In the experiment, the Nao successfully cleaned up the environment and placed all objects into their containers. In this scenario, the robot could not observe the complete scene from its initial position. To complete its world knowledge before cleaning up all objects, it had to execute the *detect_objects* action after delivering the cube. The initial planning time in this scenario was 22 seconds, with an additional planning time of 56 seconds after the second object detection. The complete execution of the experiment took 530 seconds.

IX. Conclusion

In this paper, we presented a modular approach that enables humanoid robots to solve complex mobile manipulation tasks autonomously. The core of the system consists of a high-level planner that tightly integrates perception, world modeling, action planning, navigation, and mobile manipulation as well as a monitoring component that verifies object and robot states for successful task execution. We build upon the Temporal Fast Downward Planner (TFD/M) for symbolic planning and extend it for pick-and-place tasks including navigation with humanoid robots.

We illustrated the performance of our system in experiments with a Nao humanoid cleaning up cluttered scenes in home-like environments. The robot first estimates the locations of the objects to be cleaned up using its onboard sensors. Then it plans an action sequence to pick up and deliver all objects to their target location while

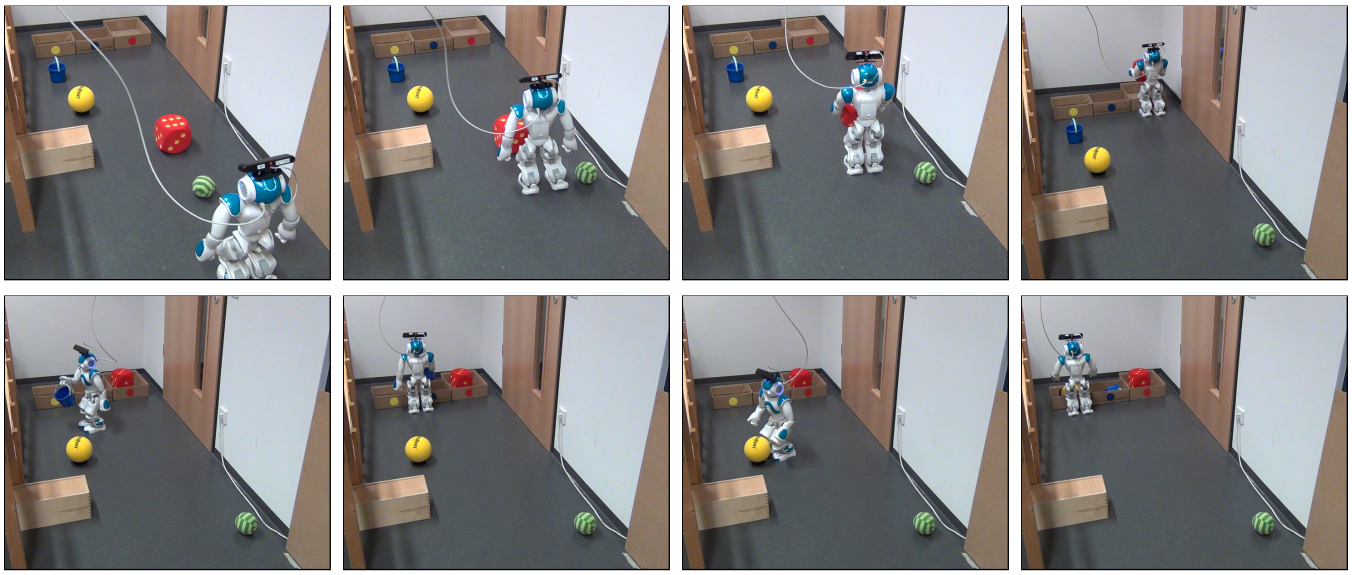


Fig. 8. A Nao humanoid cleans up a cluttered environment using our approach. The robot first has to move the small ball aside, and then pick up the red cube. After placing it into a container, the robot can clean up the remainder of the scene. A video of this sequence accompanies the paper submission.

moving clutter out of its way and avoiding collisions with static items. By continually monitoring the outcome of planned actions, our approach leads to reliable execution of the planned actions. The planning time of the symbolic planner scales exponentially with the number of objects. Specialized heuristics could speed up solving the clean-up task considered here. However, with the employed general symbolic planner, we can easily solve other tasks such as delivery by adapting the planning domain. Our system is highly modular, which means that individual components can be interchanged, e.g., different techniques for object perception, grasping, or motion planning can be employed. Core components of our approach are available open source at `http://wiki.ros.org/humanoid_navigation` and `http://wiki.ros.org/symbolic_planning`.

In future work, we plan to extend the abilities of the robot towards more sophisticated object recognition [17], grasping unknown objects [18], and whole-body motion planning for object manipulation [19].

References

[1] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," in *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. AAAI Press, 2009.

[2] M. Stilman and J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal on Humanoid Robotics*, vol. 2, no. 4, pp. 479–504, December 2005.

[3] M. Stilman, K. Nishiwaki, S. Kagami, and J. Kuffner, "Planning and executing navigation among movable obstacles," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[4] H.-N. Wu, M. Levihn, and M. Stilman, "Navigation among movable obstacles in unknown environments," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[5] M. Levihn, J. Scholz, and M. Stilman, "Hierarchical decision theoretic planning for navigation among movable obstacles," in *Workshop on the Algorithmic Foundations of Robotics*, 2012.

[6] M. Levihn, K. Nishiwaki, S. Kagami, and M. Stilman, "Autonomous environment manipulation to assist humanoid locomotion," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

[7] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation," in *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2010.

[8] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *Int. Journal of Robotics Research (IJRR)*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[9] A. Gaschler, R. P. A. Petrick, M. Giuliani, M. Rickert, and A. Knoll, "KVP: A knowledge of volumes approach to robot task planning," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 202–208.

[10] C. Dornhege and A. Hertle, "Integrated symbolic planning in the tidyup-robot project," in *AAAI Spring Symposium - Designing Intelligent Robots: Reintegrating AI II*, 2013.

[11] C. Dornhege, A. Hertle, and B. Nebel, "Lazy evaluation and subsumption caching for search-based integrated task and motion planning," in *Proceedings of the IROS Workshop on AI-Based Robotics*, 2013.

[12] B. Nebel, C. Dornhege, and A. Hertle, "How much does a household robot need to know in order to tidy up your home?" in *AAAI Workshop on Intelligent Robotic Systems*, 2013.

[13] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *Journal of AI Research (JAIR)*, vol. 20, pp. 61–124, 2003.

[14] A. Hornung, S. Oßwald, D. Maier, and M. Bennewitz, "Monte Carlo localization for humanoid robot navigation in complex indoor environments," *International Journal of Humanoid Robotics*, vol. 11, no. 2, 2014.

[15] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012.

[16] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2004.

[17] W. Wohlkinger, A. A. Buchaca, R. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[18] J. Stückler, R. Steffens, D. Holz, and S. Behnke, "Efficient 3d object perception and grasp planning for mobile manipulation in domestic environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1106 – 1115, 2013.

[19] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.