2017 IEEE-RAS 17th International Conference on
Humanoid Robotics (Humanoids)
Birmingham, UK, November 15-17, 2017

Efficient Coverage of 3D Environments with Humanoid Robots Using Inverse Reachability Maps

Stefan Oßwald Philipp Karkowski Maren Bennewitz

Abstract— **Covering a known 3D environment with a robot's camera is a commonly required task, for example in inspection and surveillance, mapping, or object search applications. In addition to the problem of finding a complete and efficient set of view points for covering the whole environment, humanoid robots also need to observe balance, energy, and kinematic constraints for reaching the desired view poses. In this paper, we approach this high-dimensional planning problem by introducing a novel inverse reachability map representation that can be used for fast pose generation and combine it with a next-best-view algorithm. We implemented our approach in ROS and tested it with a Nao robot on both simulated and real-world scenes. The experiments show that our approach enables the humanoid to efficiently cover room-sized environments with its camera.**

I. Introduction

You probably know this problem: Your TV remote, glasses, and car keys keep vanishing, especially when you are in a hurry. A personal assistance robot can help you out by searching for the lost object in your apartment. Humanoid robots are particularly well suited for this task, as they can bend over to look into boxes like the robot in Fig. 1, or bend down to peek below tables and chairs. In this paper, we present an approach for calculating an efficient plan for covering a known environment with the camera of a humanoid robot by integrating view point planning with whole-body motion planning. We assume that the robot already has a 3D map of the environment and the robot's task is then to completely cover relevant areas with its camera as efficiently as possible.

Our approach adapts the next-best-view algorithm for 3D coverage by Dornhege *et al.* [1] to meet the requirements of humanoid robots. The original approach has been used in RoboCup Rescue disaster scenes on tracked vehicles with cameras mounted on a robotic arm. Humanoids with head-mounted cameras, in contrast, have a much smaller reachable volume to place the camera due to the kinematic limitations of the robot. Additionally, humanoid robots need to consider walking and pose stability, energy consumption, and the risk of overheating joints when planning view poses for observing the scene. Due to the high degree of freedom, planning whole-body postures while searching next-best-view poses is computationally highly expensive. Hence, we propose to use pre-computed inverse reachability maps that can be queried efficiently during the view point planning stage. In a first step, our approach generates promising view poses by casting rays from surfaces and sampling candidate camera poses for free-space voxels where many of these rays pass through.

All authors are with the Humanoid Robots Lab, University of Bonn, 53113 Bonn, Germany, {sosswald,philkark,maren}@cs.uni-bonn.de.

Fig. 1. Nao inspecting an environment. The robot bends over to peek into a box for completing the task of completely covering the environment with its camera, e.g., for finding objects.

Afterwards, we use the inverse reachability map to evaluate possible whole-body configurations to reach the views. As a result, we get a set of camera poses that cover the environment and formulate a travelling salesman problem to find the best connecting tour for the humanoid.

Our experiments in simulation and real-world scenarios with a Nao robot show that our system enables the humanoid to successfully and efficiently inspect home-like environments covering all interesting surfaces. The proposed coverage planning system can be applied to object search as well as to similar problems including inspection or surveillance tasks, and re-mapping the environment to incorporate changes.

II. Related Work

Finding view points from where a whole known or unknown scene can be observed is a well-known, challenging problem in both robotics and computer graphics. A large number of applications need to solve this problem, including autonomous exploration, autonomous scanning and reconstruction of 3D objects, and coverage and surveillance tasks.

The optimization problem of finding the minimum number of viewing points required for observing a known environment has been formulated as the *art gallery problem*, which asks for the positions where guards or CCTV cameras have to be placed for monitoring an art gallery with a polygonal floor plan in 2D or a polyhedral model in 3D. The art gallery problem is known to be NP-hard and APX-hard even in 2D environments [2].

978-1-5386-4677-9/17/$31.00 ©2017 IEEE

Stasse *et al.* [3] and Foissotte *et al.* [4], [5] presented a two-step approach for exploration and coverage with a humanoid robot. In the first step, a next-best-view algorithm is used for finding suitable view poses. In the second step, a posture generator tries to find postures to reach the desired view poses. The two steps are alternated in a greedy iterative scheme. Separating view pose planning and pose planning has the disadvantage that collision checks, stability constraints, and energy optimization cannot be considered while optimizing the view points. Generating a pose for every candidate view point is infeasible. We overcome this problem by pre-computing an inverse reachability map that can be queried fast enough to be used in the view point planning stage.

In the past, several variations of next-best-view algorithms for finding a good sequences of view points to observe a scene have been proposed. Bissmarck *et al.* [6] published a run-time comparison of some existing solutions. Next-best-view algorithms have also been successfully applied to find view points for 3D reconstruction using in-hand manipulation [7], using cameras mounted on robotic arms with a fixed base [8], and using unmanned aerial vehicles [9]. Our scenario, however, requires a humanoid robot to walk around in the environment, which makes planning more difficult as balancing constraints and pose optimization have to be considered. In contrast to approaches that reduce the problem complexity by limiting view point candidates to convex hulls [10] or bounding spheres [11] surrounding the objects of interest, we sample candidate view poses in the whole volume that the robot can reach.

While we focus on the task of covering a known environment completely, the general framework can also be used for autonomous exploration as in the work of Dornhege and Kleiner [12]. In the autonomous exploration task, covering the known surfaces of the environment is replaced by covering the *frontiers* between known and unknown regions and the information gain of a view point is estimated based on the size of the unknown voids in the field of view. Daudelin and Campbell [13] propose a probabilistic extension of the work by Isler *et al.* [14], which consider the information gain for each cell in the vicinity of frontier surfaces for computing the next best view.

In our previous work [15], we introduced an approach for speeding up exploration tasks by exploiting background knowledge. Based on a topological graph provided by the user, the robot computes a global exploration strategy using a travelling salesman problem solver. This global strategy can be combined with a local exploration strategy determined with the approach present in this paper.

Burget and Bennewitz [16] applied inverse reachability maps for selecting suitable stance poses of a humanoid for grasping tasks. This application requires high maneuverability of the endeffector, and hence the authors use a manipulability measure based on the Jacobian matrix of the kinematic chain. For our coverage task, high maneuverability is not needed and we evaluate poses based on stability, energy consumption, and required time for reaching the whole-body pose instead.

III. Problem Description and Framework

Our goal is to completely cover a known environment with the camera of a humanoid robot. We assume that a complete 3D model of the environment is already given, e.g., generated during a previous SLAM run. The robot then has to determine a sequence of lookout poses for the camera so that all relevant regions can be covered, e.g., for the purpose of executing a search or inspection task. Thus, the goal is to find a preferably small set of viewing poses that respect the robot's kinematic limits and stability constraints and from where the whole scene can be observed. As discussed in Sec. II, the problem of finding the minimum set of view poses that cover the whole environment is known to be a hard problem on its own. Solving this problem in the context of humanoid robots introduces several constraints and long planning times due to the high number of degrees of freedom, which increase the complexity of the problem even further.

We approach this challenge by implementing a sampling-based next-best-view algorithm that has already been successfully used on tracked vehicles [1]. For efficient planning for humanoid robots, we extend this approach by pre-computing possible robot poses in an inverse reachability map that can be queried efficiently while searching for good view poses. In the following sections, we will introduce our efficient implementation of the inverse reachability map and present its applicability within a next-best-view planning algorithm.

IV. Reachability Map and Pose Evaluation

Whole-body planning for humanoid robots is a challenging problem due to the high-dimensional configuration space and due to computationally expensive constraints such as posture stability and self-collision avoidance. Planning times can be significantly reduced by pre-computing valid postures and storing them as a *reachability map* (RM). A reachability map is a volumetric representation of the poses that an endeffector can reach given that the robot's base frame (i.e., the center pose between the feet poses on the ground) is located at the origin of the RM. The RM is typically computed by sampling in the configuration space. Each cell of the RM that is marked as reachable can be annotated with one or more joint configurations for reaching the desired pose together with a cost value associated with that joint configuration. During motion planning, the planner uses the RM as a lookup table for finding a set of suitable robot configurations without having to perform expensive kinematic computations or stability and self-collision checks. The planner then only has to perform location-dependent checks such as collision checks with the environment and optimize a cost criterion.

Reachability maps have already been successfully used for grasp planning with humanoid robots [17], [16] where a 6D grasp pose is given in world coordinates and the robot has to find suitable, collision-free stance poses for reaching the desired grasp pose. Transferring this concept to our application of full coverage planning, however, needs modifications as the requirements are different. In the grasp planning application, the reachable volume of the endeffector is large, whereas in our application the endeffector is a

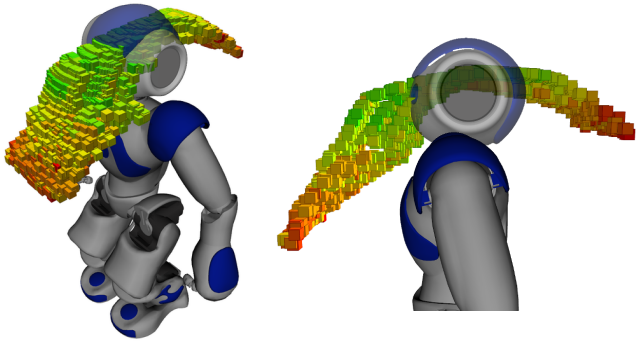

Fig. 2. Reachability map of a Nao robot. The colored boxes represent poses that the top camera mounted in the robot's head can reach given the robot's current feet positions. Green poses have low costs according to the cost function (Eq. (3)), whereas red poses with high costs should be avoided.

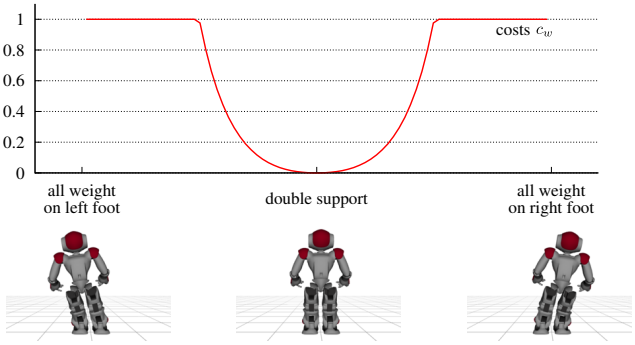

Fig. 3. Cost function for measuring pose stability based on the weight distribution between the feet. Postures in double support are preferred as they lead to bigger support polygons and are considered to be more stable than single support postures.

head-mounted camera, which typically can only move within a small and thin volume, shaped like a spherical segment (see Fig. 2 for an example). The optimization criterion in the grasp planning application is a manipulability measure. Using a tool with the robot's hand requires the ability to move the endeffector to nearby poses, e.g., for turning a screw with a screwdriver. As manipulability of the endeffector is not important for our application, we define a new cost function tailored to coverage planning that considers the stability of the pose, the time to reach the pose, and the energy consumption as humanoid robots easily overheat when resting in a stressful pose for a longer period of time.

One possibility for defining a pose stability measure is to measure the center of pressure on both feet or to calculate the zero moment point and to determine the location of the point with respect to the support polygon. When the center of pressure approaches the boundary of the support polygon, the robot posture gets unstable and small perturbations put the robot at risk of tipping over. For the Nao robots that we use during our experiments, however, the center of pressure cannot be measured reliably enough, hence, we use an approximate stability measure that compares the weight distribution on the two feet. Postures where the robot is in double support are more stable than poses where all the weight rests on one foot only. Let w_l, w_r be the weight on the left and right foot, respectively, as measured by the foot pressure sensors located in the soles of the feet. Then, we define a weight ratio

$$r = \begin{cases} \min\left(\frac{\max(w_l,w_r)}{\min(w_l,w_r)}, m\right) & \text{if } \min(w_l,w_r) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

that is the ratio of the weight distribution between the two feet clamped to the range $[1, m]$ where m is a user-defined constant, meaning that a pose is considered unstable if the weight on one foot is more than m times higher than the weight on the other foot. The ratio is symmetric with respect to both feet. In practice, a value of $m = 3$ yields reasonable results. We then define a cost function

$$c_w = \frac{(r-1)^2}{(m-1)^2} \quad (2)$$

that increases quadratically with increasing weight ratio and is normalized to the range [0,1]. See Fig. 3 for an illustration of the cost function.

To estimate the time Δt to reach the desired robot posture, we measured the time to get from a standard walking pose that the robot uses for navigation to the desired pose. For the required energy consumption, we evaluate the electric current for each joint while moving from the standard walking pose to the desired pose and back to the walking pose. Given the technical properties of the motors given in the data sheet of the robot, we compute the consumed power P integrated over the whole movement.

We then combine the three cost components to a cost function

$$c \quad = \quad k_w \cdot c_w + k_t \cdot \Delta t + k_p \cdot P \quad (3)$$

with the components defined above and linear coefficients k_w, k_t, k_p that we determined experimentally. Our view-point planning algorithm tries to find poses that cover the environment while minimizing this cost function.

We sample a large number n of robot postures q in the configuration space, execute the configuration on a robot, measure the time, power, and stability data, and compute the cost function. The result is a set $R = \{(q, c(q)) \mid i = 1 \dots n\}$ of postures with associated costs (see the visualization in Fig. 2). The individual cells of this reachability map reflect the reachable workspace of the robot and the color of the boxes in the figure shows the associated cost. If more than one configuration leads to the same pose, the costs of the lowest cost pose is shown.

V. Efficient Representation of the Inverse Reachability Map

In both the grasping application and our view-point planning application, the desired endeffector pose is implied by the task, whereas the robot's base position can be chosen freely. Hence, it is more efficient to invert the reachability map to create an *inverse reachability map* (IRM) where the endeffector is located in the origin and the cells reflect the potential locations of the base frame from where the robot can reach the desired endeffector pose. The IRM can be represented as a 6D voxel structure where the voxel

coordinates correspond to the 6D pose of the robot's base frame and each voxel contains a list of one or more joint configurations. Given a desired endeffector pose, the algorithm transforms the IRM into the world coordinate system so that the origin of the IRM matches the desired endeffector pose. The intersection of the transformed IRM with the ground plane yields a list of potential stance foot positions, which then have to be checked for collisions and optimized for the cost value.

Intersecting the two volumetric representations of the IRM and the environment model [16], however, is time-consuming and storing the IRM as a sparse 6D structure is not memory efficient. Making the assumption that the robot can only stand on horizontal planes allows for a more efficient implementation. If the robot's feet rest flat on the ground, then the roll and pitch angle of the feet relative to the endeffector frame as well as the distance between the endeffector and the feet on the vertical axis can be directly derived from the endeffector coordinates in the world frame. Hence, we propose to represent the IRM as a database instead of a volumetric representation. Each entry of the database consists of the base frame pose relative to the endeffector and a list of joint configurations to reach the endeffector pose from the given base pose with a corresponding cost value. We index the database on the roll, pitch, and z component of the base frame pose. As these coordinates can be computed directly from the desired endeffector pose under the assumption that the feet rest flat on the ground, we can access the candidate base frame poses without having to perform geometrical intersections of volumes by finding the nearest neighbor in the database. In practice, the IRM can be implemented as a k-d tree or octree that allows quick nearest-neighbor searches. For a given query, the database returns a list of candidate feet poses to reach the desired camera view pose. The algorithm then selects the configuration with the lowest costs that does not collide with the environment.

VI. Planning a Tour of Viewing Poses

For generating view points and planning a sequence of poses to cover the environment, we adopt ideas of the approach by Dornhege *et al.* [1]. Please refer to the original publication for a more detailed description including mathematical formulations of the problem and algorithm.

A. Sampling Candidate Viewing Poses

The known model of the environment is represented as an OctoMap [18]. Our algorithm first determines which occupied voxels belong to surfaces that should be covered. For our application, the ground plane does not have to be observed, so we filter it out and limit the region of interest to a user-defined bounding volume. However, our system still keeps the full OctoMap for collision checks and navigation planning.

For each occupied voxel to be observed, the algorithm casts rays starting from the occupied voxel into free space. The ray is clipped at a minimum and maximum distance from the occupied start voxel corresponding to the distance range where the robot can well observe the surface. For each free-space voxel, a counter is created that counts the number of rays traversing the voxel. If many rays pass through one voxel, this voxel is assumed to be a good lookout point as many interesting surfaces can be observed. Contrary to Dornhege's original approach, we do not sample random linear rays, but subdivide the unit sphere around the occupied voxel into 512 equally shaped conic rays. We then iterate through the cells in each cone from the tip outwards and increment the voxel counters of the traversed cells. When a collision occurs, we continue with the next cone. This approach is more systematic and a better representation of the utility of the view pose candidates, as it eliminates the systematic bias of cells near walls that get traversed more often if randomly sampled linear rays are used. We filter the traversed voxels by the height above the ground and do not consider voxels that are above or below the range where the robot's camera can be placed.

We then sort the remaining free-space voxels by decreasing utility according to the ray count. For each voxel at position (x, y, z), we sample n random 3D orientations $(\phi_i, \theta_i, \psi_i)$ to get a set of 6D camera poses $\{(x, y, z, \phi_i, \theta_i, \psi_i) \mid i = 1, \ldots, n\}$. The yaw angles ψ_i are sampled from the full range $[0, 2\pi]$. As we represent the inverse reachability map as a database indexed by (ϕ, θ, z) (see Sec. V), we can directly sample roll angles ϕ_i and pitch angles θ_i for a given z from the inverse reachability map, guaranteeing that all sampled poses are within the feasible kinematic range of the robot. For each of the n camera poses, we determine the number of surface voxels that are visible in the viewing frustum, which yields a utility value for that camera pose.

B. Determining Whole-Body Configurations

If the utility of a camera pose is above a threshold, our system determines whether there is a collision-free robot pose for reaching that view. The methods used in Dornhege's original approach are not efficient and capable enough for humanoid robots with very limited reachability ranges and constraints on stability and energy consumption. To cope with these challenges, our algorithm queries the inverse reachability map (see Sec. V) to retrieve a list of candidate whole-body configurations for reaching the given camera pose. Each configuration consists of a list of joint angles, the poses of the robot's feet relative the camera frame, and a cost term. For each of the configurations, we transform the feet poses into the world coordinate system and perform a sequence of checks to determine whether the whole-body pose is reachable:

1) Check whether the desired robot location is reachable from the robot's start location. As computing a full plan from the start location to the desired location is too computationally expensive to be executed for a large number of candidate views, we instead pre-compute a 2D reachability map once at the beginning and update the reachability map in case the environment changes. In our current implementation, we generate a 2D occupancy grid map by down-projecting the 3D model onto the map. In the resulting map, we use a region-growing algorithm

to mark all free-space cells that are definitely reachable from the robot's starting position.
2) Check if the feet poses of the configuration (including stepping safety margins) collide with the environment.
3) Check whether the full body of the robot collides with the environment. In our experiments, we use the Flexible Collision Library [19] for fast collision checks.

If a given camera pose can be reached by multiple robot configurations, we select the configuration with the lowest costs according to the cost function (see Eq. (3)). If no whole-body configuration is found, then the camera pose is unreachable and is not considered further.

C. Formulation as a Travelling Salesman Problem

The user can trade off the runtime of the view pose search versus the thoroughness of the coverage by setting the number of view pose samples and the utility thresholds, i.e., the ray count for the voxels and the number of visible surface voxels for the sampled camera poses. After sampling and evaluating camera poses for all high-utility voxels, we get a set of camera poses that cover large parts of the environment. Following Dornhege's approach [1], we partition the observed voxels by the viewing poses to determine the smallest set of viewing poses that still covers all observed voxels. This step reduces the number of poses that the robot has to navigate to and thus reduces the size of the planning problem to make it tractable.

The final step before the robot can start executing its task is to compute a tour for visiting all viewing poses, starting at the robot's current location. Dornhege [1] provides a comparison of different planning algorithms including utility-based and cost-based greedy algorithms, set cover and travelling salesman planners, and an exhaustive search. The best choice of the planning algorithm depends on the application: For object search tasks, it makes sense to start with panoramic view points where large parts of the environment are visible, as it is likely that the object can be seen from these view points and the search can be completed early. Hence, a utility-based greedy approach should be used in this scenario. If, by contrast, the task requires that all view points must be visited, for example in an inspection or mapping task, then a cost-minimizing planner is preferred. For our experiments, we choose to formulate the problem as a travelling salesman problem (TSP) on a graph and use a Lin-Kernighan heuristic solver [20] to find the shortest-path solution that visits all viewing points necessary for completely covering the environment. As Dornhege showed, the additional time required for solving a TSP in comparison to greedy approaches is outweighed by the gain in the time and energy required to execute the plan, which is especially true for humanoid robots. The nodes of the TSP graph consist of the viewing poses and the robot's start location. For each pair of nodes, we add an edge annotated by the length of the shortest path between the corresponding poses as computed by an A* planner on the 2D occupancy grid map. As we don't require the robot to return to the start location, we make the graph asymmetric by replacing the costs for travelling from any view point back to the start node by 0, meaning that the robot can "teleport" at no costs from the last visited view pose back to the start. In the resulting tour, the last edge is removed, leading to the shortest open-end path starting at the robot's current location and visiting all viewing poses.

VII. Experiments

To evaluate our approach, we conducted a series of experiments in both simulated environments and real-world settings with a Nao robot by SoftBank Robotics [21]. The robot is 58 cm tall and has 25 degrees of freedom.

A. Generating the Inverse Reachability Map

As a first step, we need to record an inverse reachability map once in the beginning for the given robot.

Any posture generator can be used to create the set of samples to be stored in the inverse reachability map, including kinesthetic teaching by a human or random sampling in a physics simulator. In our case, we used the whole-body controller provided by the manufacturer as a black-box posture generator for recording the inverse reachability map. The whole-body controller formulates the generalized inverse kinematics problem as a quadratic problem including joint limit constraints and stability criteria that constrain the center of mass to the support polygon. The controller then solves the quadratic problem in a fixed cycle of 20 ms. More details on the whole-body controller are given in the manufacturer's documentation [22]. We systematically sampled head orientations in the feasible range and torso heights between 24 cm and 32 cm and let the whole-body controller find suitable joint configurations. The configurations are evaluated and stored in the reachability map as described in Sec. IV. In our experiments, the inverse reachability map contained 1514 robot configurations in 277 database records. For more complex robots with more degrees of freedom, the pose sampling density can be reduced to keep the algorithm efficient while still covering the full configuration workspace.

B. Coverage of Simulated Environments

We tested our approach first in simulation experiments with a Nao humanoid simulated using SoftBank's Choregraphe framework. To generate footstep plans, we used the anytime search-based footstep planner by Hornung *et al.* [24]. To study our algorithm's behavior in realistic settings, we downloaded openly available models of indoor rooms and apartments provided by the Blender community and rendered the models as OctoMaps [18]. Fig. 4 shows an example scene in a bathroom based on a model from [23]. The red surfaces are the relevant environment regions that the user selected for the robot to inspect. The figure in the middle visualizes the utility map generated with the algorithm described in Sec. VI-A. Light blue voxels are traversed by many conic rays emitted from the user-selected surfaces, thus these voxels are panoramic view points from where large parts of the environment are visible. The algorithm considers these voxels first when sampling view poses. The bottom figure of Fig. 4 shows the resulting robot poses and the areas covered by the robot's camera in green. Some surfaces are unobservable for

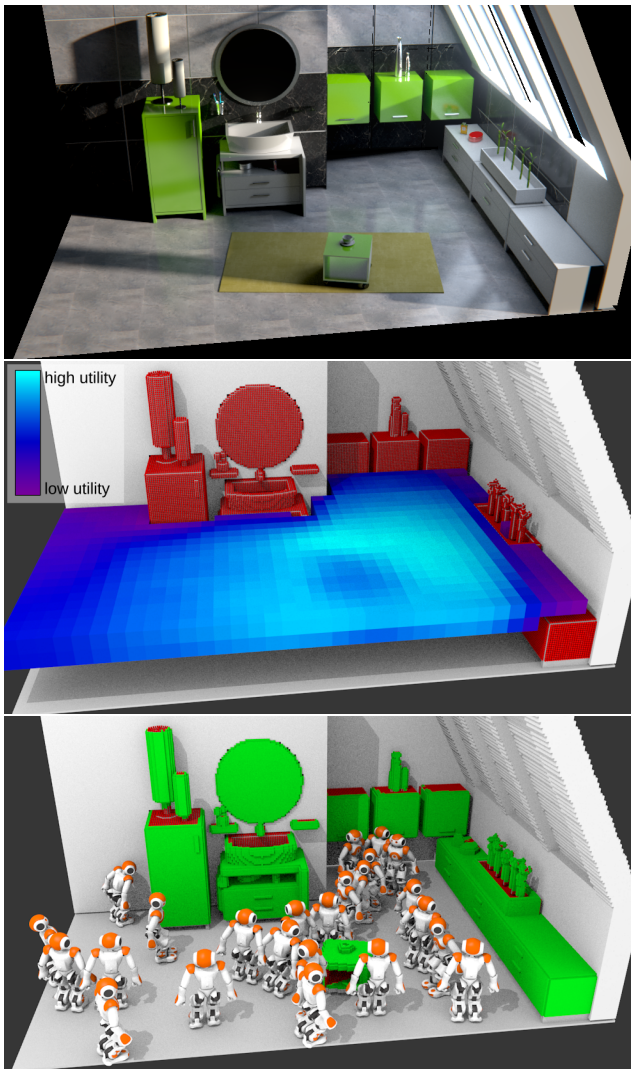

Fig. 4. Bathroom scene, model based on [23]. *Top*: Overview of the scene. *Middle*: Utility of candidate viewing poses (see Sec. VI-A). Light blue voxels indicate panoramic view points from where large parts of the scene are visible. The robot's task is to observe the red parts of the environment, whereas the gray objects are only considered for collision checking and path planning, but do not have to be observed. *Bottom*: Resulting set of poses and areas covered by the robot's camera (green). The maximum viewing range of the camera is 3 m.

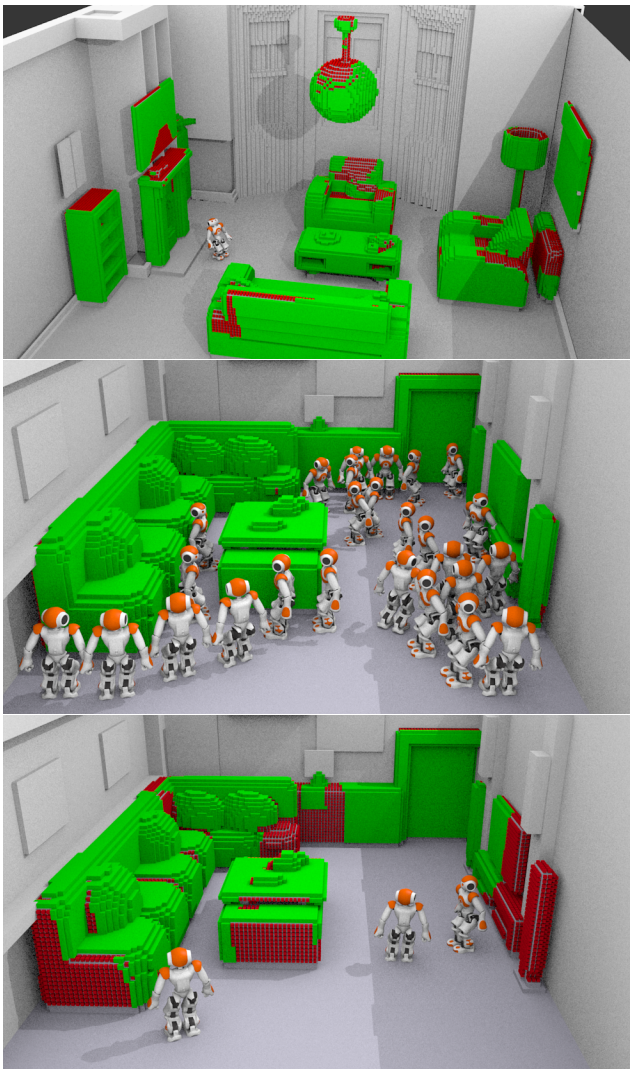

Fig. 5. Coverage results of further simulation experiments. *Top*: "The White Room" model based on [25]. Note that the robot inspects the ceiling lamp by leaning backwards and looking up. *Middle*: "Living Room" model based on [26] fully covered by 26 robot poses. *Bottom*: Same environment as above covered by only 3 high-utility robot poses. By adapting the utility threshold, the user can trade off between completeness of coverage versus the number of poses and thus the time for task completion.

the robot due to its body height, e.g., the top face of shelves.

Fig. 5 shows additional experiments in other environments. The top figure visualizes a living room scene and highlights that the robot also has to lean back and look up to inspect the ceiling lamp. The middle and bottom figure of Fig. 5 show the same living room scene covered with different numbers of view poses. By choosing the threshold for the candidate view utility appropriately, the user can trade off between completeness of coverage versus runtime. A low utility threshold (Fig. 5 middle image) leads to 26 view poses that cover all observable details of the scene. A higher threshold on the utility, in contrast, leads to a small set of high-utility poses that already cover large parts of the environment. The bottom image of Fig. 5 shows the coverage of the three highest utility poses that already cover most of the environment. In object search tasks, a planner should be used that executes these highest utility poses first, thus the robot searches for the object from panoramic view points first and proceeds with lower utility, close-up view points of smaller details until the object has been found.

C. Coverage of a Real Scene

We conducted experiments with a Nao humanoid in a real-world environment with children's toys and furniture that matches the size of the robot. Fig. 6 shows an example scenario of such an environment where the robot successfully covers all objects.

VIII. Conclusions

In this paper, we presented a framework for planning view points and full-body postures for covering a known

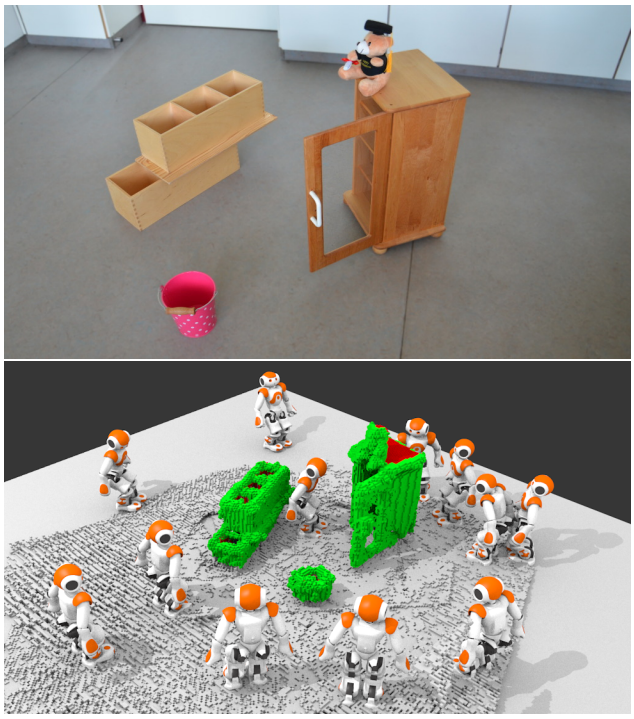

Fig. 6. Real-world experiment. *Top*: Overview of the scene. *Bottom*: Resulting set of poses and covered surfaces (green). The computed strategy contains both panoramic view points where large parts of the scene are visible and close-up view poses for peaking into the cupboard. The maximum viewing range of the camera is 1.5 m.

environment with the camera of a humanoid robot. We introduced a novel representation for inverse reachability maps that supports fast sampling and validation of robot poses. Integrating our inverse reachability map representation with a sampling-based next-best-view algorithm allows us to interleave view point planning with full-body pose planning for a humanoid robot. Our algorithm produces a set of full-body postures that are feasible and energy efficient and allow the robot to cover the whole observable 3D environment with its camera. In combination with a travelling salesman problem solver, our algorithm generates an efficient plan that can be used in a wide range of applications including inspection, surveillance, mapping, and search tasks. In future work, we intend to narrow down the interesting surfaces further, based on object affordances and geometric constraints on where searched objects can be placed.

Acknowledgment

The authors would like to thank Christian Dornhege for providing his implementation of 3D coverage search.

References

[1] C. Dornhege, A. Kleiner, and A. Kolling, "Coverage search in 3D," in *Proc. of the Int. Symp. on Safety, Security and Rescue Robotics (SSRR)*, 2013.

[2] J. O'Rourke, *Art Gallery Theorems and Algorithms*. New York, NY, USA: Oxford University Press, Inc., 1987.

[3] O. Stasse, T. Foissotte, D. Larlus, A. Kheddar, and K. Yokoi, "Treasure hunting for humanoids robot," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, ser. Workshop on Cognitive Humanoid Vision, Daejeon, South Korea, 2008.

[4] T. Foissotte, O. Stasse, A. Escande, P.-B. Wieber, and A. Kheddar, "A two-steps next-best-view algorithm for autonomous 3D object modeling by a humanoid robot," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[5] T. Foissotte, O. Stasse, P.-B. Wieber, A. Escande, and A. Kheddar, "Autonomous 3D object modeling by a humanoid using an optimization-driven next-best view formulation," *Int. Journal on Humanoid Robotics, Special issue on Cognitive Humanoid Vision*, vol. 7, no. 3, 2010.

[6] C. F. Bissmarck, M. Svensson, and G. Tolt, "Efficient algorithms for next best view evaluation," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2015.

[7] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3D object models using next best view manipulation planning," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[8] S. Kriegel, T. Bodenmüller, M. Suppa, and G. Hirzinger, "A surface-based next-best-view approach for automated 3D model completion of unknown objects," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[9] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3D exploration," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2016.

[10] E. Palazzolo and C. Stachniss, "Information-driven autonomous exploration for a vision-based MAV," in *Proc. of the ISPRS Int. Conf. on Unmanned Aerial Vehicles in Geomatics (UAV-g)*, 2017.

[11] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, "Volumetric next-best-view planning for 3D object reconstruction with positioning error," *Int. Journal of Advanced Robotics Systems*, vol. 11, no. 10, p. 159, Oct. 2014.

[12] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," *Journal of Autonomous Robots*, vol. 27, no. 6, pp. 459–468, 2013.

[13] J. Daudelin and M. Campbell, "An adaptable, probabilistic, next best view algorithm for reconstruction of unknown 3D objects," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2017.

[14] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3D reconstruction," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2016.

[15] S. Oßwald, M. Bennewitz, W. Burgard, and C. Stachniss, "Speeding-up robot exploration by exploiting background information," *IEEE Robotics and Automation Letters (RA-L)*, vol. 1, no. 2, 2016.

[16] F. Burget and M. Bennewitz, "Stance selection for humanoid grasping tasks by inverse reachability maps," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Seattle, USA, 2015.

[17] N. Vahrenkamp, D. Muth, P. Kaiser, and T. Asfour, "IK-Map: An enhanced workspace representation to support inverse kinematics solvers," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2015.

[18] 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, Feb. 2013, software available at http://octomap.github.com.

[19] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[20] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, pp. 106–130, 2000.

[21] Aldebaran Robotics. (2015) NAO documentation. [Online]. Available: http://doc.aldebaran.com/2-1/home_nao.html

[22] ——. (2015, Aug.) Whole body control - Aldebaran 2.1.4.13 documentation. [Online]. Available: http://doc.aldebaran.com/2-1/naoqi/motion/control-wholebody.html

[23] Blendswap user "cenobi". (2012) "Bathroom". Blender model, available under a Creative Commons Attribution (CC-BY 3.0) license. [Online]. Available: https://www.blendswap.com/blends/view/52486

[24] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012.

[25] J. Hardy. (2012) "The White Room". Blender model, available under a Creative Commons Attribution (CC-BY 3.0) license. [Online]. Available: https://www.blendswap.com/blends/view/41683

[26] T. Nguyễn. (2013) "Living Room". Blender model, available under a Creative Commons Zero (CC0 1.0) license. [Online]. Available: https://www.blendswap.com/blends/view/70842