Integrated Perception, Mapping, and Footstep Planning for Humanoid Navigation Among 3D Obstacles

Daniel Maier   Christian Lutz   Maren Bennewitz

Abstract— **In this paper, we present an integrated navigation system that allows humanoid robots to autonomously navigate in unknown, cluttered environments. From the data of an onboard consumer-grade depth camera, our system estimates the robot's pose to compensate for drift of odometry and maintains a heightmap representation of the environment. Based on this model, our system iteratively computes sequences of safe actions including footsteps and whole-body motions, leading the robot to target locations. Hereby, the planner chooses from a set of actions that consists of planar footsteps, step-over actions, as well as parameterized step-onto and step-down actions. To efficiently check for collisions during planning, we developed a new approach that takes into account the shape of the robot and the obstacles. As we demonstrate in experiments with a Nao humanoid, our system leads to robust navigation in cluttered environments and the robot is able to traverse highly challenging passages.**

I. Introduction

The human-like design and locomotion allows humanoid robots to step over or onto obstacles, to reach destinations only accessible by stairs or narrow passages, and to navigate through cluttered environments without colliding with objects. These abilities would make humanoid robots ideal assistants to humans, for instance in housekeeping or disaster management. However, there is a number of reasons, why up to today, we do not see such robots in practical applications. First, there are financial reasons. Humanoid robots are complex pieces of hardware and manufactured in small numbers, resulting in high prices. Second, many researchers apply navigation algorithms that represent a humanoid using a circular shape [1, 2, 3]. This model does not respect all the navigation capabilities of humanoid robots and therefore more appropriate approaches are necessary for navigation in cluttered and multi-level scenarios. Third, while some researchers focus on planning locomotion for humanoid robots, they often neglect sensing. Instead, they assume a known model of the world [4, 5], or they use external sensing systems [2, 6, 7]. Onboard sensing is, however, essential for autonomous navigation in unknown or only partially known environments. Finally, a seamless combination of individual system components including environment modeling, pose estimation, and gait generation is required for humanoids to carry out complex tasks. For all these individual aspects, promising approaches have been presented. Yet, an integrated system that combines the best solutions for all the subtasks, has not been demonstrated. Recently, Nishiwaki *et al.* [8] presented an impressive system that combines environment mapping, footstep planning, and gait control. However, for localization the system relies solely on odometry and collision checking is performed only on foot level, i.e., neglecting the body of the humanoid.

All authors are with the Humanoid Robots Lab, University of Freiburg, Germany. This work has been supported by the German Research Foundation (DFG) within the SFB/TR-8, within the Research Training Group 1103, and within the Cluster of Excellence EXC1086.

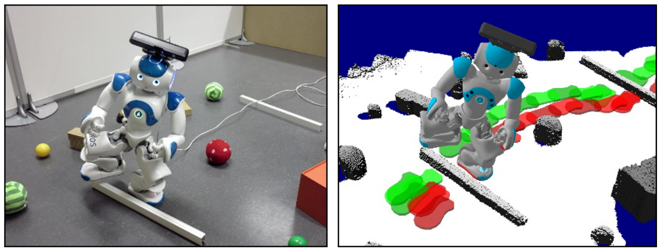
Fig. 1. Left: A Nao humanoid autonomously traversing a cluttered scene. Right: The corresponding heightmap representation of the environment that the robot generates during navigation based on data from its head-mounted depth camera. The heightmap is used for planning safe footsteps.

In this paper, we present an integrated navigation framework that combines pose estimation, mapping, and motion planning for autonomous navigation in unknown 3D environments. Our system relies only on the robot's onboard sensors, i.e., its joint encoders, an inertial measurement unit (IMU), and a head-mounted depth camera. The environment is represented as an accurate heightmap that is constructed by integrating multiple measurements over time while the humanoid navigates. Our approach performs efficient whole-body collision checking and applies traversability analysis to determine safe footprints. To robustly navigate in challenging scenes containing obstacles on the ground and narrow passages, our anytime planner computes a sequence of actions that consists of planar footsteps, step-over actions, as well as parameterized step-onto and step-down actions.

As we demonstrate in practical experiments with a Nao humanoid, our system leads to robust navigation in cluttered, multi-level scenes containing objects of various shapes and sizes. The left image of Fig. 1 shows our Nao stepping over a slat and the right image shows the corresponding heightmap constructed by integrating multiple depth camera measurements while navigating. One can see that the map closely resembles the scenario on the left and our system allows the robot to autonomously traverse the cluttered scene by planning a sequence of safe actions. Preliminary results of this work have been published in [9, 10].

II. Related Work

Autonomous biped navigation has been studied intensively in the last few years. For instance, Chestnutt *et al.* [6] investigated footstep planning among flat obstacles using A*.

Hornung *et al.* [4] have reasoned about the impact of different heuristics applied to anytime footstep planning. These approaches check for collisions only by considering rectangular footprints of the robot and do not consider volumetric obstacles. Furthermore, they neglect onboard sensing.

Perrin *et al.* [7] also investigated footstep planning and evolved it further to account for the 3D shape of the humanoid and the obstacles. They perform collision checks for the legs of the robot by precomputing swept volume approximations of the swing leg trajectories. Perrin *et al.* [11] further suggested to simplify the collision check for near real-time performance by approximating the robot's shape with a combination of three boxes. While navigating, both approaches either rely on an external motion capture system [7] to localize the robot and the obstacles, or assume known, simulated environments [11].

Other authors combine footstep planning and sensing in one system to allow for more autonomous navigation. For instance, Cupec *et al.* [12] identify obstacles in camera images to plan footsteps around them. In their approach, the authors impose constraints on the shape and appearance of obstacles and the floor. Michel *et al.* [13] presented a method to track objects in monocular images. This enabled a HRP-2 robot to accurately localize itself relative to a staircase and plan footsteps to climb it. However, the approach requires a detailed a priori model of the object. Thus, both techniques are not generally applicable for collision-free navigation in unknown environments with arbitrary objects.

More general approaches try to construct a representation of the free and occupied space in the environment while navigating. For example, Gutmann *et al.* [3, 14] maintain a labeled heightmap and a 3D occupancy grid based on data from an onboard stereo camera. The mapping system relies solely on odometry. The map representation is classified into floor, stairs, borders, tunnels, or obstacles and is used for planning discrete actions to a target location. For collision checking, the required space of each action is approximated by cylinders. The coarse resolution of the map and the approximative collision checks do not allow to plan actions such as to step over objects.

Nishiwaki *et al.* [8] utilize a tilting laser scanner mounted on a humanoid robot for environment mapping. While navigating, their robot takes 3D scans of the area in front. The laser point clouds are binned into cells of a heightmap which is used for judging the quality of possible footprint locations and planning a sequence of safe stepping positions. Because this approach also relies solely on odometry for pose estimation, old data is deleted from the map to reduce artifacts resulting from accumulated errors. Furthermore, collision checks are only performed for the footprints, i.e., they disregard the body of the humanoid.

In our previous work [1], we presented an efficient navigation system for humanoid robots. It utilizes a consumer-level depth camera for pose estimation in a given 3D model of the environment and for mapping of unknown objects. For planning paths, the approach projects the 3D map onto the ground plane and checks for collisions with a circular approximation of the robot's shape. Accordingly, actions such as stepping over or onto objects cannot be considered. In the current work, we lift these limitations and, furthermore, construct accurate heightmaps on which planning is performed.

Recently, simultaneous localization and mapping (SLAM) systems that operate on RGB-D camera data have been presented [15, 16]. These approaches are concerned with global consistency of the constructed maps and are computationally demanding. In the presented work, we do not intend to solve similar problems but focus on maintaining a locally consistent and accurate map that can be used for 3D footstep planning and whole-body collision checking. Newcombe *et al.* [17] presented Kinect Fusion, an approach for dense surface modeling and pose tracking with RGB-D cameras. While in general the system provides very impressive results, it relies on the presence of sufficient variation in depth. In preliminary experiments, we observed that the system is not robust enough in typical robot navigation scenarios with more sparse clutter on a dominant floor plane.

III. Pose Estimation

Many approaches to biped navigation rely only on odometric information to obtain the pose of the robot [3, 8, 18]. However, odometry is prone to drift over time and abrupt errors due to slipping of the robot's feet. To increase the accuracy of the pose estimation, we hence combine odometry with depth measurements.

Our approach computes a transformation T^t_{corr} that corrects the current odometry estimate T^t_{odom} at time t such that it is consistent with the previous estimate with respect to the sensor readings from the depth camera. Here, T^t_{odom} defines the 6D transformation from a fixed origin to the current pose $(x, y, z, \varphi, \psi, \theta)$ of the depth sensor consisting of the 3D position and the roll, pitch, and yaw angles. To estimate T^t_{odom}, the system maintains the 6D transform to the current stance foot and assumes it to be fixed while the swing foot moves. Using forward kinematics, the poses of all the robot's joints and links, including the depth camera, can easily be computed. The transform to the stance foot is updated from forward kinematics, whenever the swing foot becomes the new stance foot. In this way, our approach is also able to estimate the robot's pose even if the height changes, e.g., when the robot steps onto objects. To compensate for small errors in the joint encoder readings or slightly inclined terrain, our system uses the pitch and roll measurements from an onboard IMU. Consequently, the algorithm rotates the pose estimate around the current stance foot such that the reference frame of the IMU as computed by forward kinematics aligns with the readings from the IMU. The IMU is typically installed in the humanoid's chest.

As stated, estimating the pose T^t_{odom} in this way is sensible to accumulating errors. Hence, we use a depth camera to improve the estimate by seeking a transform

$$T^t_{\text{corr}} = \underset{T'}{\arg\min} \sum_i \left\| T^{t-1}_{\text{corr}}\, T^{t-1}_{\text{odom}} \mathbf{q}_i - T' T^t_{\text{odom}} \mathbf{p}_i \right\|^2 , \quad (1)$$

where $\mathbf{q}_i \leftrightarrow \mathbf{p}_i$ are corresponding points in two consecutive point clouds from the depth camera at time $t-1$ and t. The corrected pose estimate is given by the concatenation of the transforms $T^t_{\text{corr}}\, T^t_{\text{odom}}$.

Since there are no exact correspondences in real sensor data, we estimate the transform between two point clouds by using the Generalized-ICP (GICP) algorithm [19]. Thereby, the GICP is initialized with the odometry estimate. Since the ground plane typically dominates the scene and thus also the alignment process, we filter out points belonging to the ground plane prior to applying GICP.

IV. Environment Representation

To represent the environment and plan collision-free motions, our system maintains a high-resolution heightmap that is learned from depth camera data. Each cell c of the map stores a height value h_c and a variance σ^2_c. The variance represents the uncertainty about the height of each cell resulting from small pose estimation errors and sensor noise. Hence, we interpret $\mathcal{N}(h_c(t), \sigma^2_c(t))$ as the belief about the height of c at time t. To update the map, the points from the current point cloud are binned into the cells of the heightmap. Let z_c be the maximum over the z-coordinates of all observed points falling into a cell c. Our approach then updates the belief about the height of c from $\mathcal{N}(h_c(t), \sigma^2_c(t))$ and the observation z_c using a Kalman filter, assuming a state model with no underlying dynamics [20]. Hence, we assign

$$h_c(t+1) = \frac{1}{\sigma^2_c(t) + \sigma^2_z}\left(\sigma^2_z\, h_c(t) + \sigma^2_c(t)\, z_c\right), \quad (2)$$

$$\sigma^2_c(t+1) = \frac{1}{\sigma^2_c(t) + \sigma^2_z}\, \sigma^2_z\, \sigma^2_c(t), \quad (3)$$

where σ^2_z represents the uncertainty of the observation. Due to the employed depth camera as sensor, σ^2_z is best modeled proportional to the quadratic distance from the sensor to the observed point [21].

V. Footstep Planning for 3D Environments

A. State Representation and Transition

In our planning system, a state $s = (\hat{x}, \hat{y}, \hat{\theta}, f)$ is expressed by a location $(\hat{x}, \hat{y})$, an orientation $\hat{\theta}$, and $f \in \{\text{left}, \text{right}\}$ indicating whether the left or right foot is the stance foot. The height z_s of a state s is determined uniquely from the heightmap as the average over the height values covered by the robot's footprint at s, and hence not part of the state space for planning. Further, the state space is discretized over $\hat{x}$, $\hat{y}$, and $\hat{\theta}$ and represented by a sparse graph.

For planning motions, we consider a set of discrete actions $\mathcal{A}$. For an action $a \in \mathcal{A}$, $a(s)$ describes the transition $s \xrightarrow{a} s'$ from a state s to its successor s'. An action a is consequently parameterized by $(\Delta\hat{x}, \Delta\hat{y}, \Delta\hat{\theta}, f)$, where f indicates the stance foot for the action and the remaining parameters the displacement of the swing foot relative to f. Furthermore, each action is parameterized over an interval $[\Delta z_{\text{min}}, \Delta z_{\text{max}}]$ that describes the admissible height differences from a state to its successor when executing this action. For planar footsteps, it is $\Delta z_{\text{min}} = \Delta z_{\text{max}} = 0$.

B. Safe Actions

During planning, our system evaluates which of the actions $a \in \mathcal{A}$ can be executed safely from a given state s. Our approach first checks whether the resulting footprint $s' = a(s)$ is accessible, i.e., whether the corresponding surface of the map is flat and horizontal. Accordingly, we compute the difference between the minimum and maximum values in the heightmap under the footprint and check whether it lies within a threshold (implied by the hardware).

Additionally, an action a is only allowed if the height difference $\Delta h(s,a) = z_{a(s)} - z_s$ from s to its successor $a(s)$ lies within the limits Δz^a_{min} and Δz^a_{max} associated with a. Hence, we require the following inequality to hold:

$$\Delta z^a_{\text{min}} \leq \Delta h(s,a) \leq \Delta z^a_{\text{max}} \quad (4)$$

Finally, for motion planning in three-dimensional environments it is important to check whether the motions are free of collisions, i.e., not only the footprints as is typically done in footstep planning approaches. Our approach to whole-body collision checking is described in the following.

C. Whole-Body Collision Checking

We propose a new representation called inverse heightmap (IHM), which is computed for each action. An IHM is a grid, centered at the stance foot, that stores for each cell the minimum height relative to the stance foot of any body part that falls into this cell while executing an action. It is constructed from an animated 3D model of the robot. The bottom image in Fig. 2 shows an example of an IHM for a step-over motion, along with the 3D model used for generating the IHM (top row). The green volume is a 3D visualization of the IHM and similar to a swept volume. However, for efficient collision-checking, we only consider its projection in the IHM.

To evaluate whether it is safe to execute an action a at a state s, our system first aligns the corresponding IHM^a at the pose defined in s with height z_s by an affine transformation. Then, a simple comparison on the height values is used to decide whether a is safe:

$$\forall (u,v) \in \text{IHM}^a : i_{(u,v)} + z_s > h_{(u,v)}. \quad (5)$$

Here, z_s is the height of state s, $i_{(u,v)}$ is the height stored in cell (u,v), and $h_{(u,v)}$ is the corresponding value in the heightmap. Because of this simple decision criterion, IHMs are an efficient way to perform whole-body collision checks. Note that they can be precomputed for all actions $a \in \mathcal{A}$.

This approach, however, assumes that it is known where the current swing foot comes from. Therefore, we assume that every footstep passes through a predefined via point configuration, similar to [22]. The IHMs consequently consist of the downwards phase of an action from the via point configuration to the double support phase and of the upwards phase to the via point of the other foot.

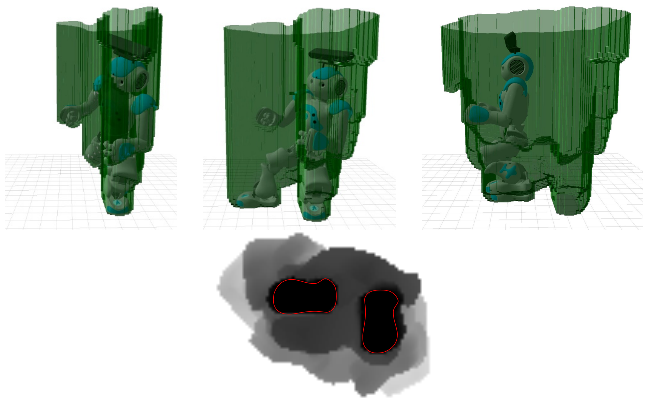

Fig. 2. Generation of an inverse heightmap (IHM) for a step-over action. The top three images are snapshots of a 3D model of the robot executing the action along with the volume covered by the motion (green). The bottom image shows the resulting IHM (the darker the lower) that corresponds to the projection of the volume onto the ground plane. The footprints of the robot's stance feet are outlined in red for reference.

D. *Footstep Planning with ARA**

Our planner searches for the optimal solution with respect to the time needed to reach a target state and plans a sequence of safe actions that ensure collision-freeness of the whole body. To plan footstep sequences, we rely on Anytime Repairing A* (ARA*), which is an efficient anytime variant of A*, i.e., it finds an initial solution as fast as possible, while guaranteeing a bound on its suboptimality. Afterwards, the algorithm tries to refine the solution in the remaining time [4, 23]. This type of algorithm has two advantages. First, it is goal directed and, second, an initial, valid solution is computed fast. The latter is especially useful if the plan has to be updated often, e.g., due to updates of the environment representation.[1]

For the goal-directedness of ARA*, a heuristic has to inform the algorithm about the estimated costs to the goal from any state in the search space. Our approach aims at minimizing the travel time, thus, the heuristic predicts the remaining time to reach the goal location. We considered two different heuristics and compared their performance in the experimental evaluation in Sec.VII-A. To be able to compute the optimal path, the heuristic needs to be admissible, i.e., it may not overestimate the true costs. For both heuristics, we obtain the predicted remaining time $h(s)$ from a state s to the goal by the estimated remaining distance $d(s)$ divided by the velocity v_{max} of the fastest action the robot can perform:

$$h(s) = d(s)/v_{\text{max}} \tag{6}$$

1) Euclidean Distance Path Cost Heuristic: One simple heuristic for obtaining the remaining distance $d(s)$ to the goal is the Euclidean distance. It promotes expanding states on the straight line to the goal. This heuristic is clearly admissible. On the down-side, it is often a poor approximation in cluttered environments where detours are inevitable or step-over actions are associated with high costs [4].

[1]Note that in our experiments, ARA* yielded betted results than R* [4, 24] due to our well-designed heuristics and since the scenarios are not dominated by local minima.

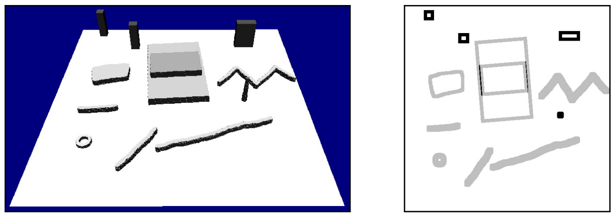

Fig. 3. The left image shows an example heightmap, and the right image the corresponding traversability costs used to generate the Dijkstra heuristic.

2) Dijkstra Path Cost Heuristic: We also developed a more informed heuristic that better approximates the true distance in presence of obstacles. The heuristic is based on Dijkstra's shortest path algorithm. In particular, we construct a graph from the heightmap, where each node represents a $(\hat{x}, \hat{y})$-location of the discrete state space and is associated with a corresponding height value from the heightmap. The edges of the graph represent the eight-neighborhood of the nodes.

Our algorithm assigns traversal costs to each edge according to the difference of the height values in its neighborhood. If the height difference exceeds the maximum step height of the robot, the edge is considered non-traversable and infinite costs are assigned. Edges in a planar neighborhood are considered planar as well and low costs are assigned. All other edges correspond to a change in the elevation where the robot could step up, down, or over. To account for the additional time to execute these actions, our approach assigns higher costs to such elevating edges. Afterwards a full Dijkstra search is performed on the graph to compute the metric distance $d(s)$ of each node s to the goal state. Fig. 3 gives an example for the traversability classification of a heightmap. Here, dark corresponds to non-traversable, bright to planar, and gray to elevating edges. The particular costs of the edges depend on the target hardware platform and should not overestimate the true costs. Otherwise the heuristic would be inadmissible.

VI. Action Set for the Nao Humanoid

In this section, we describe the action set for the Nao humanoid that we used during the experimental evaluation. Nao is a small-sized humanoid developed by Aldebaran Robotics. It is 58 cm in height, weighs 5.2 kg and has 25 degrees of freedom. With the provided walking controller, the swing foot can be placed at most 8 cm to the front and 16 cm to the side and the peak elevation is 4 cm. The size of the robot's feet is approximately 16 cm×9 cm. From these numbers, it is clear that Nao is not able to step over, onto, or down from obstacles using the standard motion controller. The discrete set consisting of 12 basic footsteps that we chose from the possible motions of the standard controller is shown in Fig. 4(a).

Using kinesthetic teaching, we designed motions that allow the robot to overcome these limitations. A special motion, the so-called T-step, where the feet are placed at an angle of 90° (see Fig. 4(b)), is the basis for the other actions. Our motivation for this action is to exploit the larger

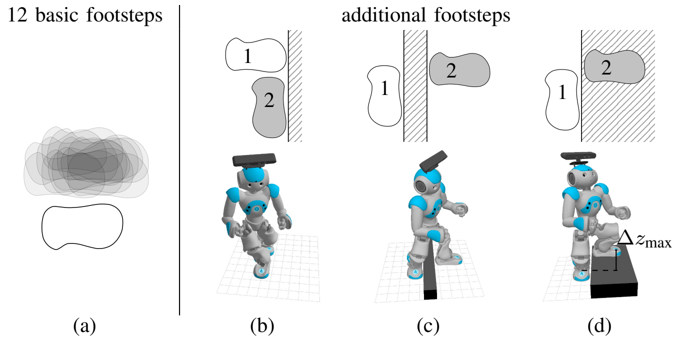

Fig. 4. Footsteps set for Nao. (a) Basic planar footsteps, (b) T-step, (c) Step-over action, (d) Step-onto/down action. Step-over and Step-onto/down actions are preceded by a T-step. All actions are also mirrored for the other foot.

lateral foot displacement while moving forward. From this pose, the robot can perform a step-over action to overcome obstacles with a height and width of 6 cm (see Fig. 4(c)). Furthermore, from the T-step the robot is able to step onto or down from obstacles. Fig. 4(d) illustrates the robot stepping onto an obstacle exemplary for this motion. The motion is similar to the step-over action but the swing foot is placed closer to the stance foot and at a different height. The height is adjusted online using inverse kinematics based on the value in the heightmap. The height difference relative to the stance foot must be in the interval $[\Delta z_{\min}, \Delta z_{\max}]$ for that action as defined in Sec. V-B. In our implementation, we set $\Delta z_{\min} = -7$ cm and $\Delta z_{\max} = 7$ cm. All motions also exist in a mirrored version for the other stance foot. Thus, we used 16 actions in total per foot for our experiments.

VII. Experimental Evaluation

In this section, we present the results from a thorough evaluation of our system. We first evaluate our 3D planner quantitatively in experiments with simulated heightmaps. Afterwards, we demonstrate the capabilities of our navigation system in a series of real-world experiments. All experiments were carried out with a small-sized Nao humanoid with a head-mounted ASUS Xtion depth camera.

In the experiments presented in the following, the robot takes a point cloud with its camera every second step (generally, it is also possible to increase the number of steps between two consecutive measurements). The robot waits about 0.5 s before recording the point clouds to reduce disturbances caused by its shaking motion and delays between the joint encoder readings and the depth camera data. To obtain a larger field of view, the robot takes two depth images with its camera facing left and right and combines them to one large point cloud. Here, the camera is pitched down by 50° and the yaw angle is +17° and -17°, respectively. Our planner treats unknown areas as free space to allow for planning into unknown areas, which often occurs when only onboard sensor data is used. While executing the planned motions, the robot actively looks in direction of the next unknown area to update the heightmap. After each new measurement the robot checks if the planned path is still valid. If not, the robot re-plans the path. A video demonstrating the system is available online at `http://youtu.be/g2NZ_EasJv0`.

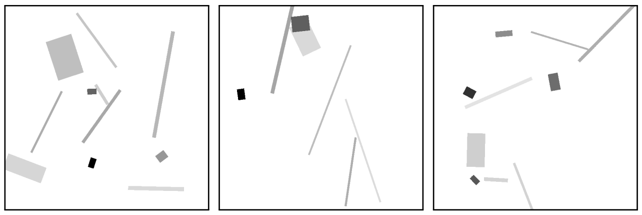

Fig. 5. Three randomly sampled maps consisting of bars, platforms, and blocks used to quantitatively evaluate our 3D planner. The level of the color gray hereby indicates the height of the cells (the darker, the higher).

A. *Quantitative Evaluation of the 3D Planner*

To evaluate our planner and the two different heuristics quantitatively, we randomly generated ten different heightmaps. The sampled maps are all of size 2.5 m $\times$ 2.5 m with a resolution of 4 mm and contain obstacles such as bars, platforms, and blocks of varying width, length, and height. Three example heightmaps are shown in Fig. 5. We sampled start and goal locations uniformly such that they were collision-free and their distance was between 1.5 m and 3.0 m. We then used our motion planner to generate safe trajectories on a computer with an Intel Core i5 3.1 GHz CPU. The initial suboptimality bound for the planner was set to 8 in all experiments. All 100 planning problems could be solved by ARA* within the given time limits of 5 s and 10 s, respectively. As can be seen from Table I, the extended Dijkstra heuristic leads to more efficient solutions that are closer to the optimal path compared to the straight-line Euclidean distance heuristic (significant at 95% level). Shown are the mean and standard deviation of the path cost suboptimality (path costs divided by the costs of the optimal solution). On average, it took 97 s to compute the optimal plan, whereas our anytime algorithm generates first, valid solutions within less than a second on average and afterwards improves the found plan. Furthermore, we observed that the whole-body collision checks using inverse heightmaps consumed only 27.1% $\pm$ 2.7% of the planning time.

We also evaluated the planning performance when considering the 3D structure and the extended action set compared to 2D footstep planning [4]. To this end, we applied ARA* using only the set of planar footsteps shown in Fig. 4(a) to the same set of 100 planning problems. The 2D planner was only able to solve 91% of the planning problems within the limit of 10 s. In most of the generated maps, our new 3D planner outperformed the 2D variant in terms of paths costs since the latter one had to choose detours around obstacles. In some simpler scenarios, however, the generated solutions of the 2D planner were superior to the ones of the 3D planner. The reason is that the 3D planner has to additionally perform whole-body collision checks and has a higher branching factor so that the 2D planner could expand more states and found better solutions within the given time limit.

B. *Evaluation of Localization and Mapping*

The following experiment is designed to evaluate the state estimation accuracy of our approach in terms of both localization and mapping performance. We tracked the pose of the robot with an optical motion capture system from

TABLE I

QUANTITATIVE EVALUATION OF THE MOTION PLANNER.

Heuristic t-Limit	Dijkstra 10 s	Euclid 10 s	Dijkstra 5 s	Euclid 5 s
Suboptimality	1.04±0.05	1.13±0.14	1.08±0.10	1.19±0.19
$t_{\text{init\_sol}}$ [s]	0.73±0.83	0.61±0.54	0.71±0.80	0.58±0.52

Motion Analysis while the robot traversed the course shown in the top image of Fig. 6. The bottom image shows the corresponding heightmap learned by the robot during navigation using our approach. The path traversed by the robot is indicated by the footprints and lead over a bar. As one can see, the map closely resembles the actual structure of the scenario shown in the top image.

Using the motion capture system, we measured the accumulated error between the tracked pose and the pose estimated by our approach and between the tracked pose and the odometry. Fig. 7 shows the results in terms of the planar translational and rotational error, plotted over the actual traveled distance. We observed that the accumulated drift of the pose estimate was 0.21 m in *xy*-direction over the whole trajectory of 3.47 m. Hence our system drifts approximately 5.9 cm per traveled meter. Analogously, the accumulated drift in the *yaw*-angle was 3.46° and hence, on average our system drifts 0.99° per meter. For odometry, we noted an accumulated drift of 0.53 m in *xy*-direction and 6.10° in the *yaw*-angle. Hence, the average drift is 15.3 cm and 1.76° per traveled meter, respectively. Thus, our approach clearly outperforms the pure odometry estimate obtained from forward kinematics of the measured leg joint angles. Furthermore, the experiment illustrates that our localization method is able to reduce the drift of the pose estimate and allows for constructing an accurate environment map.

We also measured the time for aligning the point clouds and updating the heightmap. On average, a heightmap update from a combined point cloud took 0.07 s ± 0.02 s. Combining the point cloud from the left and the right view, and aligning the resulting point cloud to the previous combined point cloud with GICP took 0.41 s ± 0.22 s.

C. *Parameterized Stepping Over and Onto Motions*

In the remaining two experiments, we present qualitative results of our framework with a Nao humanoid. Fig. 8 shows an experiment in which the robot climbed two stairs up and down again. No model of the stair was used, not even the height of the stairs was known beforehand. The heightmap was constructed online and the height of the footsteps was computed according to this representation.

D. *Traversing Narrow Passages*

The final experiment demonstrates the advantage of applying whole-body collision checks. Fig. 9 shows a scenario where a Nao humanoid needed to traverse a passage that was so narrow that the humanoid could not walk through facing forwards without its arms colliding with the obstacles. Consequently, our motion planner computed a path where the robot traversed the passage sideways and without collisions.

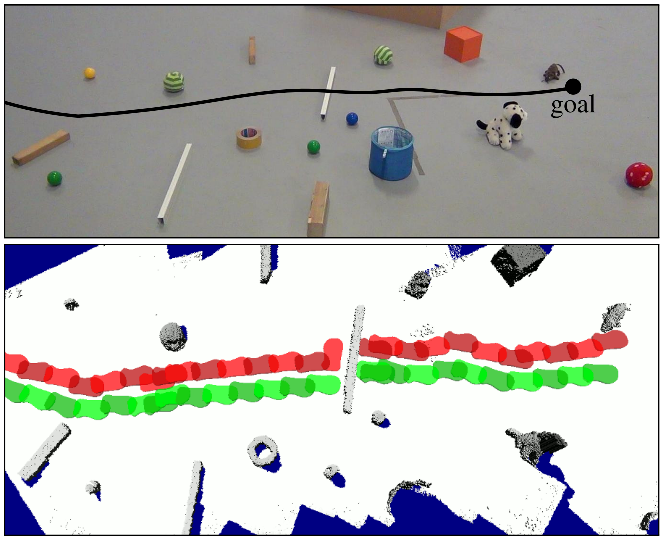

Fig. 6. Top: Cluttered scenario autonomously traversed by the robot from left to right for evaluating the state estimation accuracy. The black line shows the path traversed by the robot (manually drawn). Bottom: The corresponding heightmap learned during navigation by our approach along with the path traversed by the robot. As can be seen, the environment is represented highly accurate. The dark area correspond to unobserved areas.

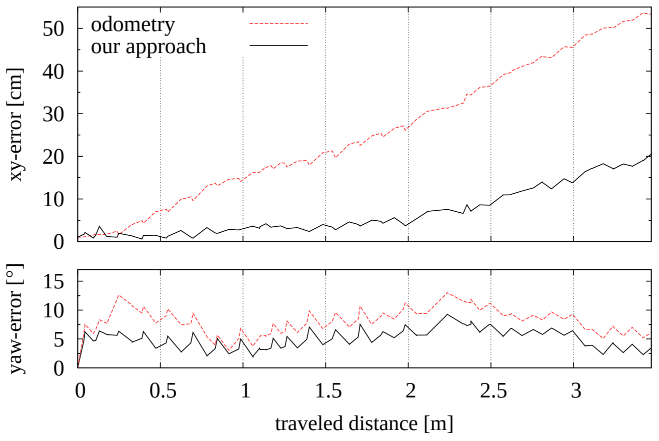

Fig. 7. Localization accuracy relative to the ground truth from an optical motion capture system. Our approach clearly outperforms the odometry estimate and leads to an accurate pose estimate during navigation.

As the lower body fitted through in a forward direction, motion planners that consider only the feet or legs of the robot for collision checks might have found a solution leading to a collision during the execution of the plan.

VIII. CONCLUSIONS

In this paper, we presented an integrated approach that enables a humanoid robot to navigate in previously unknown, cluttered environments. Our system includes incremental pose estimation based on odometry and point cloud alignment using GICP, mapping of the environment using an accurate heightmap representation, anytime footstep planning, and whole-body collision checking using inverse heightmaps. To the best of our knowledge, this is the first system that combines these techniques in a unique framework.

As the experiments with a Nao humanoid show, our technique to pose estimation clearly outperforms the odom-

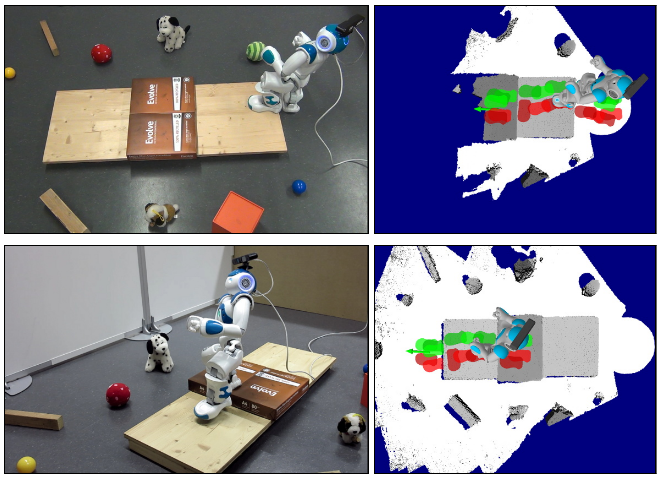

Fig. 8. Nao climbing up and down steps by carrying out parameterized step onto and down actions. The right column shows the heightmap representation at the time of the image on the left along with the current footstep plan and the pose estimate.

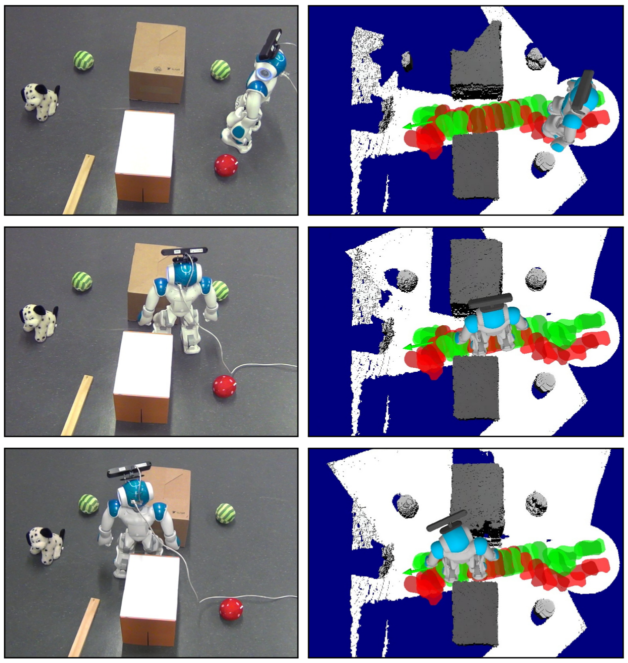

Fig. 9. Traversing a narrow passage. The Nao humanoid cannot walk through the passage facing forwards because its arms would collide with the obstacles. Our planner performs whole-body collision checks and thus computes a solution where the robot traverses the passage sideways.

etry estimate and allows for the construction of accurate heightmaps. Based on this, our robot plans and robustly executes sequences of actions that include stepping over and climbing up or down obstacles as well as passing through narrow passages. Our approach to planning and collision-checking based on a learned heightmap representation can be generally applied to any humanoid robot.

References

[1] 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.

[2] 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.

[3] J.-S. Gutmann, M. Fukuchi, and M. Fujita. Real-time path planning for humanoid robot navigation. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2005.

[4] 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.

[5] K. Hauser, T. Bretl, and J.-C. Latombe. Non-gaited humanoid locomotion planning. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2005.

[6] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade. Footstep planning for the Honda ASIMO humanoid. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2005.

[7] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida. Fast humanoid robot collision-free footstep planning using swept volume approximations. *IEEE Transactions on Robotics (T-RO)*, 28(2), 2012.

[8] K. Nishiwaki, J. Chestnutt, and S. Kagami. Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor. *Int. Journal of Robotics Research (IJRR)*, 2012.

[9] A. Hornung, D. Maier, and M. Bennewitz. Search-based footstep planning. In *Proc. of the ICRA Workshop on Progress and Open Problems in Motion Planning and Navigation for Humanoid*, 2013.

[10] D. Maier, C. Lutz, and M. Bennewitz. Autonomous biped navigation through clutter. In *Proc. of the RSS Workshop on Robots in Clutter: Preparing Robots for the Real World*, 2013.

[11] N. Perrin, O. Stasse, F. Lamiraux, Y. J. Kim, and D. Manocha. Real-time footstep planning for humanoid robots among 3D obstacles using a hybrid bounding box. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[12] R. Cupec, G. Schmidt, and O. Lorch. Experiments in vision-guided robot walking in a structured scenario. In *IEEE Int. Symp. on Industrial Electronics*, 2005.

[13] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade. GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[14] J.-S. Gutmann, M. Fukuchi, and M. Fujita. 3D perception and environment map generation for humanoid robot navigation. *Int. Journal of Robotics Research (IJRR)*, 27(10):1117–1134, 2008.

[15] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[16] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *Int. Journal of Robotics Research (IJRR)*, 31(5):647–663, 2012.

[17] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. of the IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, 2011.

[18] J. Chestnutt, Y. Takaoka, K. Suga, K. Nishiwaki, J. Kuffner, and S. Kagami. Biped navigation in rough environments using on-board sensing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[19] A. Segal, D. Hähnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[20] P. Pfaff, R. Triebel, and W. Burgard. An efficient extension of elevation maps for outdoor terrain mapping and loop closing. *Int. Journal of Robotics Research (IJRR)*, 2007.

[21] K. Khoshelham and S. Oude Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors: Journal on the Science and Technology of Sensors and Biosensors*, 12:1437–1454, 2012.

[22] J. Kuffner Jr, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Footstep planning among obstacles for biped robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[23] 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.

[24] M. Likhachev and A. Stentz. R* search. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2008.