Humanoid Navigation with Dynamic Footstep Plans

Johannes Garimort Armin Hornung Maren Bennewitz

***Abstract*— Humanoid robots possess the capability of stepping over or onto objects, which distinguishes them from wheeled robots. When planning paths for humanoids, one therefore should consider an intelligent placement of footsteps instead of choosing detours around obstacles. In this paper, we present an approach to optimal footstep planning for humanoid robots. Since changes in the environment may appear and a humanoid may deviate from its originally planned path due to imprecise motion execution or slippage on the ground, the robot might be forced to dynamically revise its plans. Thus, efficient methods for planning and replanning are needed to quickly adapt the footstep paths to new situations. We formulate the problem of footstep planning so that it can be solved with the incremental heuristic search method D* Lite and present our extensions, including continuous footstep locations and efficient collision checking for footsteps. In experiments in simulation and with a real Nao humanoid, we demonstrate the effectiveness of the footstep plans computed and revised by our method. Additionally, we evaluate different footstep sets and heuristics to identify the ones leading to the best performance in terms of path quality and planning time. Our D* Lite algorithm for footstep planning is available as open source implementation.**

I. Introduction

Compared to wheeled robots, the greater flexibility of humanoid robots allows for unique capabilities such as climbing stairs and stepping over or onto objects instead of bypassing them. However, exploiting these capabilities during path planning is far more complex because many degrees of freedom have to be controlled. Planning whole body motions for navigation in the real world is computationally not feasible yet [1], [2]. A common solution is to use predefined motions to walk on a sequence of footstep locations, which reduces the problem to planning a sequence of collision-free footsteps (e.g., [3], [4], [5], [6]).

In this paper, we also adopt the approach of footstep planning to efficiently search for optimal paths in environments containing planar obstacles, uneven ground, holes, or general locations which have to be avoided. In contrast to previous approaches which use A* search for planning footstep paths, we apply D* Lite [7], an incremental heuristic search method for computing optimal paths. The advantage of D* Lite is that it allows for efficient replanning. This is particularly relevant for humanoid robots which regularly have to correct deviations from an initial path because of accumulated motion drift from foot slippage and joint backlash. Additionally, D* Lite enables the robot to reuse information from previous searches when it has to revise its footstep plan according to changes in the environment.

All authors are with the Humanoid Robots Lab at the Department of Computer Science, University of Freiburg, Germany.

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

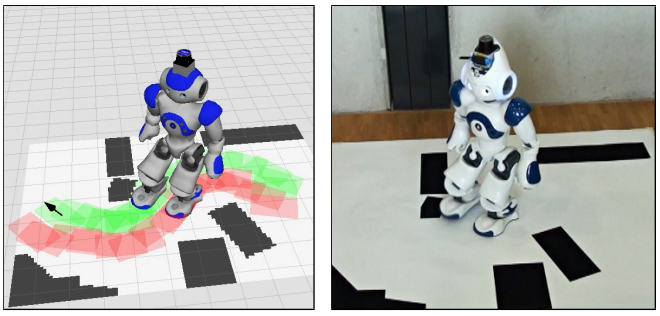


Fig. 1. A footstep path in an environment with planar obstacles (*left*) is executed by our Nao humanoid (*right*), carefully stepping around the black obstacles. To efficiently adapt to changes in the environment, we use the D* Lite algorithm [7] and extend it to planning with continuous footsteps.

We formulate the problem of footstep planning so that it can be solved with D* Lite and present our extensions, including efficient collision checking for footsteps and continuous footstep locations. We thoroughly evaluate different sets of possible footsteps and heuristics guiding the search, and discuss the effects on planning time and path quality in various environments. In experiments carried out in simulation and with a *Nao* humanoid (see Fig. 1), we demonstrate the effectiveness of the footstep plans computed by our D* Lite footstep planning algorithm. Our footstep planning framework is available as C++ source code[1].

II. Related Work

Research in the area of navigation planning for humanoids amongst obstacles differs in how many details are considered during planning and how far the planning horizon is chosen. Several techniques concentrate only on generating the next movements of the robot. The authors focus on reactive collision avoidance or stair climbing behaviors and do not consider global navigation tasks. For example, Cupec *et al.* [8] and Yagi and Lumelsky [9] use predefined walking primitives to locally plan paths. Okada *et al.* [10] apply a local planner that generates footsteps along a straight line. A drawback of these local planning methods is that they are prone to end up in local minima on the way to the goal.

Other approaches plan a global 2D path for the robot and then locally determine footsteps to follow the trajectory [11], [12]. Furthermore, approaches have been proposed that seriously constrain the possible footstep locations of the robots or use heuristics to simplify the planning problem. Examples are the techniques presented by Ayaz *et al.* [5] and Cupec and Schmidt [6]. As a result, the set of problems which can

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

be solved by these methods is limited and the resulting plans are not optimal.

Gutmann *et al.* [13] as well as Candido *et al.* [14] presented planning methods for humanoids that are based on different motion primitives. These approaches do not take into account that humanoid robots can also avoid obstacles by adapting their footsteps.

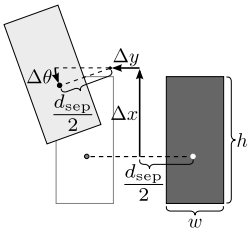
Chestnutt *et al.* [3] proposed to plan the whole path entirely on footstep basis using A*. To constrain the search, the authors defined a set of seven reference actions. Later, the authors extended their technique so that, depending on the terrain below the foot location, different footsteps close to the reference action can be chosen in case there is no valid position in the original footstep set [4]. In contrast to these approaches which use A* with a special inadmissible heuristic to avoid starting from scratch when replanning, we apply D* Lite [7] to dynamically replan the path when necessary.

Some authors developed techniques to generate whole body motions which also contain non-upright gaits. For example, Kanehiro *et al.* [15] proposed to locally plan whole body motions using a ZMP-based pattern generator that can change the waist height. Hauser *et al.* [16] presented a probabilistic planner that first samples contacts of predesignated parts of the robot's body and points of the terrain. Afterwards, a probabilistic roadmap method is applied to connect feasible robot configurations. In a later work, the authors make use of motion primitives to guide the search and generate high-quality motions [1]. While the results are impressive, so far only simulation experiments have been presented and it remains unclear how the correct motion primitive is selected. Kanoun *et al.* [2] combine footstep planning with inverse kinematic to reach kinematic goals. The whole body is controlled so that complex tasks, such as picking up an object from the ground, can be carried out. The general problem of planning whole body motions is that these approaches require long planning times and are only applicable to short paths due to their computational complexity.

The incremental search methods D* and D* Lite have been widely used for robot navigation. For example, variants were employed by the winning team of the DARPA Grand Challenge [17] and on the surface of Mars [18]. In this paper, we present the first extension of D* Lite to planning global footstep paths for humanoids.

III. Humanoid Robot Platform

We use the *Nao* humanoid robot by Aldebaran Robotics for our practical experiments (see Fig 1). Nao is 58 cm tall, weighs 4.8 kg, and has 25 degrees of freedom. For proprioception, Nao is equipped with Hall effect sensors to measure the angle of each joint and an inertial measurement unit to yield an estimate about the its attitude. In addition to other sensors such as ultrasound and two cameras, our humanoid is equipped with a Hokuyo URG-04LX laser range finder mounted in a modified head. By interpreting the data of the laser range finder and the robot's proprioception, we can account for motion drift and accurately determine the robot's pose in a 3D world model [19].



Parameter	Value / Range
d_{sep}	9.5 cm
w	9.0 cm
h	16.0 cm
Δx	$[-4.0, 4.0]$ cm
Δy	$[-1.0, 4.0]$ cm
$\Delta\theta$	$[-2.9, 20.0]^\circ$

Fig. 2. Footstep model and parameterization of the Nao.

Nao's walking engine uses a linear inverted pendulum model to generate stable gaits [20]. The walking engine can be controlled with omnidirectional velocities. While the corresponding motions can be immediately executed, e.g., to follow a two-dimensional path, this requires larger safety margins to obstacles because there is no direct control over the exact movement of the robot and thus the placement of its feet. However, the walking engine also allows to walk on a given sequence of foot placements or footsteps (API version: 1.6). These are defined by a displacement vector $(\Delta x, \Delta y, \Delta\theta)$, as illustrated in Fig. 2. The neutral position $(0, 0, 0)$ results in the walking foot being placed next to the supporting foot in parallel, with their origins (projected ankle joints) the distance d_{sep} apart. The parameterization in Fig. 2 ensures that there are no self-collisions while walking. We will use this footstep model throughout this work. Note that any given 2D foot displacement with orientation can be transformed into this representation.

IV. D* Lite

In our work, we build upon the incremental heuristic search algorithm D* Lite [7]. This search algorithm extends the heuristic search of the well-known A* algorithm by reusing information from previous searches to allow for an efficient replanning.

A. Overview

A* is a search algorithm for finite graphs, commonly used for path planning in known 2D grid maps. The algorithm employs a best-first search from a starting state s_{start} to the goal s_{goal}, which is guided by the heuristic h. For each pair of states $s, s' \in S$, $h(s, s')$ yields an estimate of the costs of traversing the graph from s to s'. The more informed the heuristic is of the actual costs in the environment, the more efficient A* can search since the number of expanded states is reduced.

D* Lite extends A* so that it can be used for efficient incremental searches. Hereby, the algorithm continually searches for shortest paths in the environment while the current starting state progresses along the path and the edge costs may change arbitrarily. This can be due to revised estimates of the robot's position in the environment (e.g., from a localization algorithm), changed positions of non-static obstacles, or detections of new obstacles. The incremental search of D* Lite is hereby orders of magnitude more efficient than repeatedly planning a new path from scratch with A* [7].

B. Efficient Replanning

For efficient replanning from changing starting states, D* Lite searches in the reverse order from the goal state s_{goal} to the current state s_{start}. The initial search corresponds to a complete A* search, as there is no incremental information available for replanning. Afterwards, D* Lite maintains the costs $g(s)$ for all visited states as an estimation of the actual costs $g^*(s)$ of the optimal path from state s to s_{goal}. If all $g(s)$ are consistent with $g^*(s)$, i.e., $g(s) = g^*(s)$, the optimal path can be extracted by recursively selecting

$$s' = \min_{s' \in Succ(s)} (g(s) + c(s, s')) \tag{1}$$

from s_{start} on. Here, $c(s, s')$ denotes the costs of traversing from state s to s' and $Succ(s) \subseteq S$ contains all states reachable from s. Whenever cost values in the environment change, the estimates $g(s)$ need to be made consistent with $g^*(s)$ again. Instead of adjusting all $g(s)$ on every change, D* Lite changes only relevant states. For this purpose, D* Lite maintains a priority queue which only contains those states whose costs are not consistent. While processing the queue and expanding states, D* Lite updates the estimated costs of the optimal path through those states connecting s_{start} and s_{goal}. Heuristics h are used which estimate the costs from each state to s_{start}. Similarly, the costs of the optimal path are updated after edge costs changed.

Just as in A*, the heuristic function $h(s, s')$ plays an important role in D* Lite as it guides the search from s_{goal} towards states in the direction of s_{start}. In D* Lite, it is required that for all $s, s', s'' \in S$:

$$h(s, s') \geq 0 \tag{2}$$

$$h(s, s'') \leq h(s, s') + h(s', s'') \tag{3}$$

$$h(s, s') \leq c^*(s, s') \,, \tag{4}$$

where $c^*(s, s')$ denotes the true costs from s to s' of the optimal transition. These requirements are stricter than for A* and allow for efficient cost updates of states in the priority queue as explained in the following.

D* Lite maintains lower bounds of the optimal path costs through a state to the goal in the priority queue. When replanning has to be invoked and, thus, the starting state has to be set to the robot's current pose, the algorithm does not need to change the costs of each state in the priority queue. Instead, it temporarily stores the cost change resulting from the different starting state s'_{start} of the planning problem. This cost change $h(s'_{\text{start}}, s_{\text{start}})$ is identical for all states in the priority queue with respect to maintaining the lower bounds. Thus, adding $h(s'_{\text{start}}, s_{\text{start}})$ to states which are later on inserted into the queue results in efficient replanning.

V. Footstep Planning with D* Lite

We chose D* Lite for its flexibility and efficiency and adapted it to footstep planning for humanoid robots. Its replanning capabilities can be exploited when the humanoid seriously deviates from the computed path during execution, when non-static obstacles change their locations, or when bad terrain locations are observed along the way.

In the following, we will formulate the problem of footstep planning for humanoid robots so that it can be solved with the incremental heuristic search of D* Lite. For a given finite set of possible footsteps, the planner will output the optimal sequence of footsteps, if there exists a solution. The start and goal states are hereby given as global 2D poses with orientations.

A. Environment Model

We use a 2D grid map consisting of equally-sized grid cells that are marked as either free or occupied, but do not require the robot to use this grid discretization for its footstep locations. From this 2D map, we then compute a distance map containing the Euclidean distance to the closest obstacle for each cell. We use this distance map to compute step costs and to check for collisions.

B. State and Transition Model

A single footstep is parameterized by the selection of a leg to move (left or right) and the displacement of the foot relative to the supporting foot, as described in Sec. III. Under the constraint that the robot needs to move the left and the right leg alternating and that the footsteps for both feet are symmetric, a footstep action a can be completely parameterized by

$$a = (\Delta x, \Delta y, \Delta\theta) \,, \tag{5}$$

relative to the position of the current supporting foot.

Usually, a planner has a fixed set of footstep actions $F = \{a_1, \ldots, a_n\}$ available. When expanding a state s, all successive states s' are determined by applying the transitions t of all actions $a \in F$: $s' = t(s, a)$ and discarding invalid states which would lead to collisions with obstacles.

The robot's state in the context of footstep planning can be described by the global position and orientation of its supporting foot

$$s = (x, y, \theta) \,, \tag{6}$$

alternating between the right and left foot. The costs of a state transition $s' = t(s, a)$ are given by the transition costs $c(s, s')$, which we model as

$$c(s, s') = \|(x, y), (x', y')\| + k + d(s') \tag{7}$$

for $s = (x, y, \theta)$ and $s' = (x', y', \theta')$. k are constant costs associated to executing one step, thereby penalizing paths with a higher number of steps. $d(s')$ denotes the costs of the state s' based on the obstacle distance map: Footsteps closer to obstacles have higher costs assigned. We here assume that the effort of changing the orientation of a footstep can be neglected compared to the Euclidean distance the step covers.

C. Collision Checking

In order to validate a possible footstep, the planner needs to check if the footstep collides with an obstacle in the environment. Because this validation needs to be performed for all successors of an expanded state, it should be as efficient as possible. Simply assessing the distance to the nearest obstacle would not allow footstep configurations

close to obstacles since the shape of the robot's footprint is rectangular. Therefore, we apply an efficient collision check in the distance map similar to the method suggested by Sprunk *et al.* for collision checking of a rectangular wheeled robot [21]. Hereby, the rectangular footstep shape is recursively subdivided based on its circumcircle and incircle.

D. Heuristic Functions

As explained in Sec. IV, the heuristic function $h(s)$ guides the search for a plan, effectively speeding up the process. For footstep planning with humanoid robots, we propose the following three heuristics which we extensively compare in various environments in Sec. VI-A. Note that because of the backwards search of D* Lite, $h(s)$ estimates the costs from a state $s = (x, y, \theta)$ to $s_{\text{start}} = (x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}})$. In the following equations, ω_1 and ω_2 are scaling factors for the Euclidean and angular distances.

1) Euclidean distance and number of steps:

$$h_1(s) = \omega_1 \left\| (x, y), (x_{\text{start}}, y_{\text{start}}) \right\| + k\, \mathcal{S}_1(s, s_{\text{start}}) \quad (8)$$

$\mathcal{S}_1$ estimates the number of expected footsteps based on the straight-line distance, which is then multiplied with the constant costs per step k (cf. Eq. (7)).

2) Euclidean and angular distance with number of steps:

$$h_2(s) = \omega_1 \left\| (x, y), (x_{\text{start}}, y_{\text{start}}) \right\| + k\, \mathcal{S}_1(s, s_{\text{start}}) + \omega_2 \left| \theta - \theta_{\text{start}} \right| \quad (9)$$

In addition to the Euclidean distance, h_2 is informed of the change in orientation. This helps to focus on states which minimize orientational changes.

3) Length of the 2D path and number of steps:

$$h_3(s) = \omega_1\, \mathcal{D}(s, s_{\text{start}}) + k\, \mathcal{S}_2(s, s_{\text{start}}) \quad (10)$$

$\mathcal{D}$ corresponds to the length of a 2D path, which is planned with a conventional grid-based path planner on the 2D map with enlarged obstacles according to the incircle of the robot's foot (similar to [3]). $\mathcal{S}_2$ then estimates the number of expected footsteps along this 2D path. Compared to h_1 and h_2, h_3 is better informed of the environment since local minima during the search are more likely to be avoided.

h_2 and h_3 are potentially inadmissible, violating Eq. (4), because there are no orientational costs in c as in h_2, and because the 2D path of h_3 does not take into account the capability of stepping over obstacles. This may potentially affect the length of the resulting path when a sub-optimal path is found first. In our experiments, however, all heuristics lead to optimal paths.

E. Planning with Continuous Footstep Locations

While D* Lite works only for a finite set of states, the state space S of all global footstep positions is continuous. During the search, we have to continuously compare states for equality to identify new states which have not been visited before. Here, we apply an approximative comparison of two global footstep locations for equality. Thus, we can use finite sets of footstep actions to plan in a continuous state space instead of discretizing it.

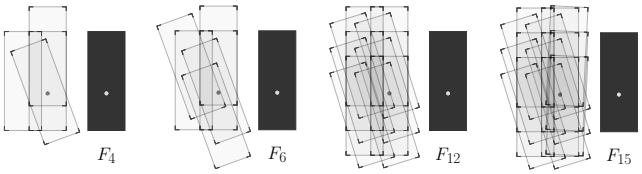


Fig. 3. Footstep sets with 4, 6, 12, and 15 different footsteps for the Nao humanoid, shown here relative to the right supporting foot (black).

Fig. 4. 2D obstacle maps of the evaluated scenarios. Each map covers an area of 1.5 m × 1.5 m at 0.01 m resolution. Planar obstacles are marked in black.

Since D* Lite plans in the reverse order from the goal with a finite set of footsteps, the desired state s_{start} may not be exactly reached. However, it is sufficient to reach any state within the motion range of the robot, the robot only has to adjust the first stepping motion then.

When replanning is invoked due to a map update, the affected states in terms of footstep locations already inserted in the priority queue have to be determined. By establishing a mapping between 2D map grid cells and visited states, i.e., footsteps locations which lie on them, we can efficiently update states in affected map regions.

VI. Experiments

Both the heuristics and the set of available footsteps F influence the planning behavior and results. The more informed the heuristic is of the actual costs, the more efficient the search can be guided. With a too coarse discretization of the robot's motion range into footsteps, the planner might fail in certain scenarios or compute longer plans, while too many actions might lead to a larger search space and inefficient planning.

Throughout the statistic evaluation, we compare given footstep sets (Fig. 3) and the previously introduced heuristics in three different environments. The environments in Fig. 4 all cover an area of 1.5 m × 1.5 m at 1 cm resolution and pose different challenges for a footstep planner. In this work, we assume that a 2D grid of the environment is given and updated when obstacle positions change. For evaluation, we randomly sample 20 collision-free start and goal poses with a straight-line distance of approximately 0.7 m. We only evaluate experiments, which can be solved within a reasonable time in all setups currently under evaluation. In this way, we discard almost impossible configurations resulting from the sampling process.

As evaluation criteria we use the *total path costs* (sum of Eq. (7) over all successive states in the plan) and the *planning costs*. As planning time varies depending on the machine,

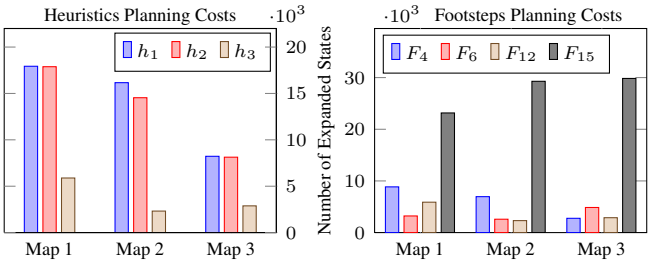


Fig. 5. *Left*: Average planning costs (in terms of number of expanded states) for heuristics in different environments. *Right:* Average planning costs for footstep sets in different environments. See text for details and statistical analysis.

we use the number of expanded states as an indicator of the planning costs. To evaluate statistical significance in the cost distributions, we apply a *paired sample t-test* ($\alpha = 0.05$) for pairs of evaluated heuristics and footstep sets with identical start and goal poses.

A. Evaluation of Different Heuristic Functions

We first compare the performance of the three heuristics introduced in Sec. V-D. The efficiency of the heuristics with respect to each other is independent of the actual choice of a footstep set, as the heuristics are not informed of the actual footstep actions. Hence, we exclusively use F_{12} for this comparison.

All three heuristics result in virtually identical path costs, which demonstrates that the potential inadmissibility of h_2 and h_3 is not a concern and all lead to optimal paths. Figure 5 (*left*) shows the planning costs averaged over the sampled start and goal poses. h_3 leads to the least number of expanded states, guiding the planner most efficiently to the goal in all environments. The paired sample t-test confirms that the difference to h_3 is statistically significant, and that h_2 is not significantly different from h_1. In summary, the heuristic h_3 results significantly fewer expanded states with no decrease in the plan quality.

B. Evaluation of Different Footstep Sets

The heuristic is set to the most informed h_3 now, and we compare the four footstep sets F_4, F_6, F_{12}, and F_{15} (Fig. 3). Note that for all $j < i : F_j \subset F_i$.

With regard to the resulting path costs, the greater flexibility of F_{12} and F_{15} yields significantly better footstep paths over the other footstep sets. However, no statistically significant difference could be detected between F_{12} and F_{15}.

According to the average number of expanded states (Fig. 5, *right*), F_{15} leads to a significant increase in planning costs compared to the other sets, while there is no significant difference between the others. Thus, we can conclude that the flexibility of the footstep set F_{12} yields efficient plans in terms of path costs, with no additional planning costs. A larger footstep set only increases the planning dimensionality without a significant benefit for the path quality.

In order to generalize over hardware different from our Nao humanoid, we also investigated a larger humanoid robot with a wider motion range in our planning framework. A typical footstep set of such a robot and a resulting footstep plan is displayed in Fig. 6. The longer legs allow wider steps and a greater flexibility, e.g., to step over obstacles. This usually also results in fewer expanded states and fewer footsteps to be taken. It took 3.02 s to expand the 13 054 states for Nao's footstep plan (Fig. 7, *left*), while it took only 0.12 s to expand the 744 states for the other robot's path (Fig. 7, *right*). Evaluation was performed on a standard 2.66 GHz Desktop PC.

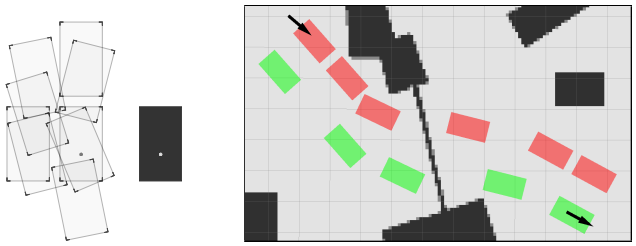


Fig. 6. A footstep set for a larger humanoid such as HRP-2 or Honda's ASIMO (*left*) allows the robot to step over obstacles (*right*).

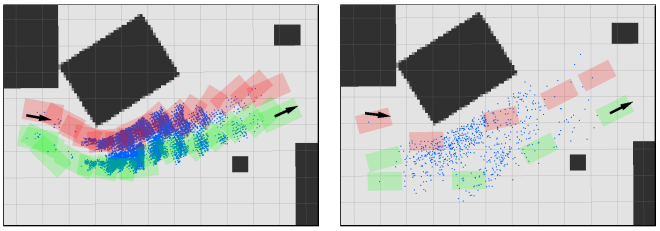


Fig. 7. Comparison of footstep path and expanded states (blue dots) between Nao's footstep set (*left*) and one of a larger robot (*right*). The wider motion range of the latter results in fewer expanded states and faster planning results (see text for details).

C. Replanning

While the humanoid follows its path to the goal, locations of obstacles can change, new obstacles can appear, or the estimated pose can deviate from the planned path. In these cases the incremental update of D* Lite allows for a fast reaction without planning the complete path anew. We evaluate the effectiveness of replanning in the scenario depicted in Fig. 8. After executing a few steps of the plan, a change in the environment is detected and the planner incrementally replans a large part of the path around the moved obstacle, reusing information from the previous plan. The initial search expanded 2966 states in 1.05 s. Replanning in the new environment expanded 956 states in 0.53 s (including

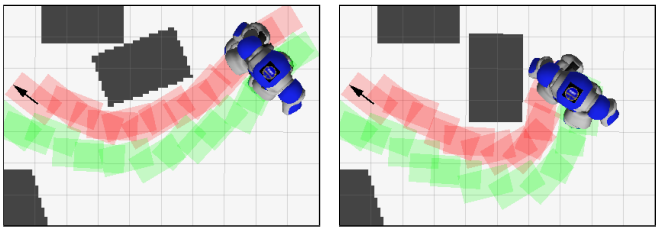


Fig. 8. While the robot traverses its initial path (*left*), the location of an obstacle changes after a few steps. Reusing previous information, a new footstep path is planned (*right*). A video of this sequence is available at `http://hrl.informatik.uni-freiburg.de`.

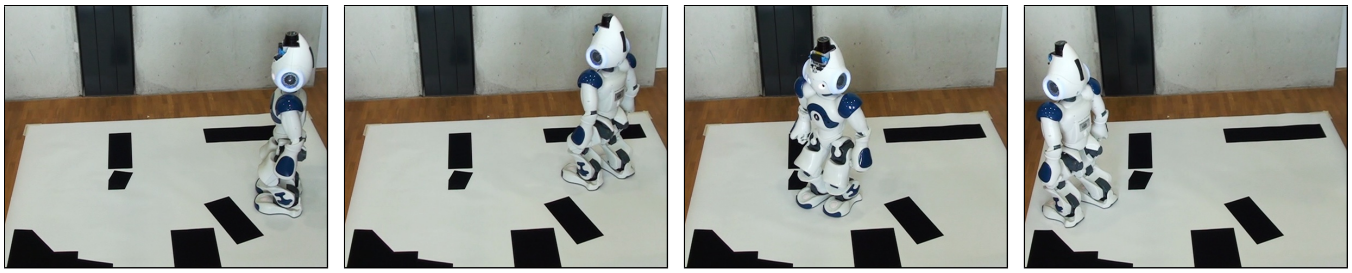


Fig. 9. Our Nao humanoid executes the footstep plan shown in Fig. 1, carefully avoiding obstacles. In this scenario, a conventional 2D path using the robot's circumcircle would lead to suboptimal results or even collisions because there is no direct control of the footstep locations. A video of this sequence is available at `http://hrl.informatik.uni-freiburg.de`.

a traversal of all states affected by the environment change). For comparison, a completely new planning cycle would have expanded 3457 states in 1.59 s in that situation.

D. Navigating with Footstep Plans

Finally, we present an experiment in which our real Nao humanoid executes a planned sequence of footsteps. Motion drift usually prevents the robot from executing a sequence of footsteps open-loop, thus requiring regular corrections based on a pose estimate from our localization system [19]. By executing the plan as a sequence of global states $\{s_1, \ldots, s_n\} \subseteq S$ instead of incremental actions a_i, the robot can correct small deviations from the original footsteps, as long as the corrected footstep is within the stepping motion range. For larger deviations, replanning with D* Lite is carried out.

Figure 9 shows our humanoid navigating in a known 2D map, executing the footstep path shown in Fig. 1 (left). As can be seen, Nao reliably navigates around obstacles to the goal location.

VII. Conclusions

Footstep planning provides an efficient way to plan motions for humanoid robots while exploiting their unique capabilities of stepping over obstacles. In this work, we presented our approach to dynamic footstep planning for humanoid navigation, building upon the incremental heuristic search of D* Lite. In experiments in simulation and on a Nao humanoid, we demonstrated the effectiveness of our methods and identified cost-efficient heuristics and footstep sets. In case of deviations from the initial plan or dynamic changes in the environment, the robot's path can be efficiently replanned by our method, thereby reusing information from previous searches. Our framework for footstep planning is released as open source at `http://www.ros.org/wiki/footstep_planner`. In the future, we plan to extend our planning method to three-dimensional environments, which enables a humanoid to step onto objects and climb stairs.

References

[1] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *Int. Journal of Robotics Research (IJRR)*, 2007.

[2] O. Kanoun, J.-P. Laumond, and E. Yoshida, "Planning foot placements for a humanoid robot: A problem of inverse kinematics," *Int. Journal of Robotics Research (IJRR)*, 2010.

[3] 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 & Automation (ICRA)*, 2005.

[4] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami, "An adaptive action model for legged navigation planning," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2007.

[5] Y. Ayaz, K. Munawar, M. Malik, A. Konno, and M. Uchiyama, "Human-like approach to footstep planning among obstacles for humanoid robots," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[6] R. Cupec and G. Schmidt, "An approach to environment modelling for biped walking robots," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[7] S. Koenig and M. Likhachev, "D* Lite," in *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2002.

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

[9] M. Yagi and V. Lumelsky, "Local on-line planning in biped robot locomotion amongst unknown obstacles," *Robotica*, vol. 18, no. 4, pp. 389–402, 2000.

[10] K. Okada, T. Ogura, A. Haneda, and M. Inaba, "Autonomous 3D walking system for a humanoid robot based on visual step recognition and 3D foot step planner," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[11] T.-Y. Li, P.-F. Chen, and P.-Z. Huang, "Motion planning for humanoid walking in a layered environment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[12] M. Elmogy, C. Habel, and J. Zhang, "Time efficient hybrid motion planning algorithm for HOAP-2 humanoid robot," in *Int. Symp. on Robotics and German Conf. on Robotics (ISR/ROBOTIK)*, 2010.

[13] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "A modular architecture for humanoid robot navigation," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2005.

[14] S. Candido, Y.-T. Kim, and S. Hutchinson, "An improved hierarchical motion planner for humanoid robots," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2008.

[15] F. Kanehiro, T. Yoshimi, S. Kajita, M. Morisawa, K. Fujiwara, K. Harada, K. K. H. Hirukawa, and F. Tomita, "Whole body locomotion planning of humanoid robots based on a 3D grid map," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[16] 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.

[17] D. Ferguson, T. Howard, and M. Likhachev, "Motion planning in urban environments: Part ii," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[18] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global planning on the mars exploration rovers: Software integration and surface testing," *J. Field Robotics*, vol. 26, no. 4, pp. 337–357, 2009.

[19] A. Hornung, K. M. Wurm, and M. Bennewitz, "Humanoid robot localization in complex indoor environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[20] D. Gouaillier, C. Collette, and C. Kilner, "Omni-directional closed-loop walk for NAO," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

[21] C. Sprunk, B. Lau, P. Pfaff, and W. Burgard, "Online kinodynamic trajectory planning for non-circular omnidirectional robots," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.