Anytime Search-Based Footstep Planning with Suboptimality Bounds

Armin Hornung Andrew Dornbush Maxim Likhachev Maren Bennewitz

***Abstract*—Efficient footstep planning for humanoid navigation through cluttered environments is still a challenging problem. Many obstacles create local minima in the search space, forcing heuristic planners such as A* to expand large areas. The goal of this work is to efficiently compute long, feasible footstep paths. For navigation, finding the optimal path initially is often not needed as it can be improved while walking. Thus, we propose anytime search-based planning using the anytime repairing A* (ARA*) and randomized A* (R*) planners. This allows to obtain efficient paths with provable suboptimality within short planning times. Opposed to completely randomized methods such as rapidly-exploring random trees (RRTs), these planners create paths that are goal-directed and guaranteed to be no more than a certain factor longer than the optimal solution. We thoroughly evaluated the planners in various scenarios using different heuristics. ARA* with the 2D Dijkstra heuristic yields fast and efficient solutions but its potential inadmissibility results in non-optimal paths for some scenarios. R*, on the other hand borrows ideas from RRTs, yields fast solutions, and is less dependent on a well-designed heuristic function. This allows it to avoid local minima and reduces the number of expanded states.**

I. Introduction

Despite the recent advances in the field of bipedal humanoid robots, efficient motion generation for navigation in cluttered environments is still a challenging problem.

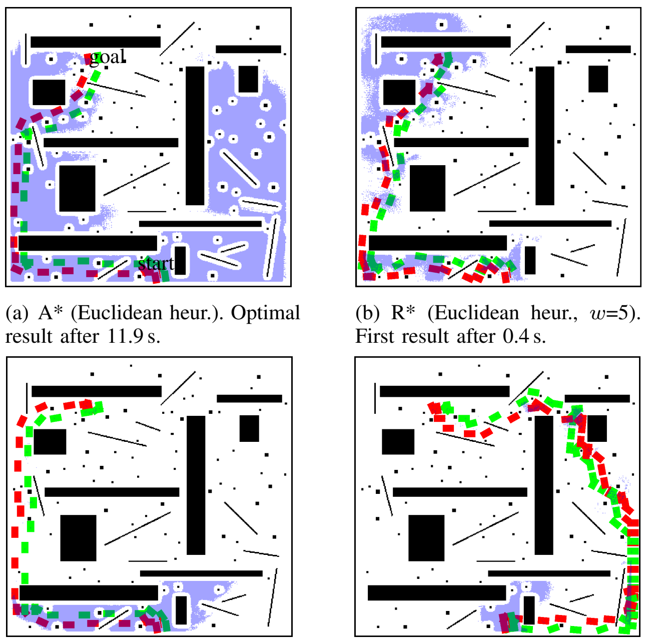
Popular approaches compute footstep trajectories given a finite set of possible footstep actions that are executable by the robot to make the search for collision-free paths tractable. A walking controller then computes joint angle trajectories to walk on the planned footsteps. The search in the space of footsteps is hereby carried out either using A*-based methods including dynamic variants (e.g., [1], [2], [3], [4]) or randomized methods [5]. While A* will find the optimal path for a given footstep parameterization and state discretization, its planning performance highly depends on the quality of the heuristic function that guides the search. Randomized methods find results fast but don't have any guarantees on the solution quality of the final path, which may be far from optimal. Furthermore, they often depend on smoothing the final path in a post-processing step [6].

In this paper, we propose to use anytime search-based planners for efficient footstep planning. In case obstacles pose local minima in the search space, these planners can efficiently create sub-optimal solutions with guarantees on the solution quality. While it is desired to fully plan the path from start to goal to decide on its feasibility, it is often not necessary to initially find the optimal path since it can be further improved while navigating. We first introduce Anytime Repairing A* (ARA*, [7]) for footstep planning and evaluate various heuristics for it. ARA* runs a series of weighted A* (wA*) searches, thereby efficiently re-using previous information. wA* inflates the heuristic by a factor w. As a second planner, which depends less on the heuristic function, we propose to use R* [8], an A* search variant that combines wA* in a local deterministic search with a randomized component, thereby trying to speed up the search. In contrast to purely randomized approaches such as rapidly-exploring random trees, R* provides probabilistic guarantees on the sub-optimality of the solution.

We present our extensions to search-based footstep planning and provide a comprehensive comparison of the planners to existing approaches (see Fig. 1). We thoroughly

A. Hornung and M. Bennewitz are with the Humanoid Robots Lab, University of Freiburg, Germany. A. Dornbush and M. Likhachev are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, USA.

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



(a) A* (Euclidean heur.). Optimal result after 11.9 s.

(b) R* (Euclidean heur., w=5). First result after 0.4 s.

(c) ARA* (Euclidean heur., w=5). First result after 2.7 s.

(d) ARA* (2D Dijkstra heur., w=5). First result after 0.7 s.

Fig. 1. Footstep planning in a densely cluttered $4 \times 4\,\mathrm{m}^2$ area. Blue (shaded) areas denote the expanded states. Computing the optimal path with A* takes a long time due to local minima, and requires expanding large areas of the state space (a). Anytime-planners provide fast solutions for realtime navigation with suboptimality bounds. R* explores the state space even with the weakly informed Euclidean distance heuristic in short time (b). ARA* depends more on the heuristic function for fast initial results (c)(d).

discuss the performance for different situations and heuristic functions. As the experiments demonstrate, we are able to plan even long footstep paths in short planning times. Due to search-based planning, the paths are directed towards the goal and have provable suboptimality bounds. ARA* requires the 2D Dijkstra path heuristic to efficiently deal with local minima such as walls blocking the path to the goal. This heuristic is potentially inadmissible since it does not consider stepping over obstacles, which leads to non-optimal results in some scenarios. R* is less dependent on the heuristic function and finds efficient results in short planning times even with the less informed but admissible Euclidean distance heuristic.

Our framework builds upon and extends the capabilities of the Search-based Planning Library (SBPL, [9]) and is available open-source as part of our ROS footstep planner at `http://www.ros.org/wiki/footstep_planner`.

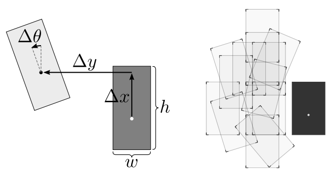
II. Related Work

In the last few years, several approaches to planning paths for humanoids have been presented. First approaches computed a local footstep path to follow a globally planned 2D path [10], [11]. The drawback here is that the globally computed 2D path may be rather inefficient since it does not consider actions to step over obstacles.

Gutmann *et al.* [12] and Candido *et al.* [13] use different motion primitives according to the type of terrain/passage and locally plan paths for regions based on the corresponding motion primitive. These approaches consider various motion behaviors but also do not adapt footstep locations to traverse objects, which may lead to sub-optimal behavior.

Recently, several approaches have been presented that use rapidly-exploring random trees (RRTs) [5] or probabilistic roadmaps (PRMs) [6] to plan motions for humanoids, including stepping over actions or highly complex and computationally expensive whole body motions. The drawback of these randomized methods is that no bounds on the quality of the found solution can be given as it is the case for A*-based planning methods. To this end, RRT* provides probabilistic convergence to an optimal solution and was recently extended for anytime motion planning [14], albeit only for a wheeled vehicle so far.

Chestnutt *et al.* [3], [4] were the first who proposed to plan footstep paths using A* search. With a set of possible footstep actions, the robot is able to traverse obstacles by stepping over them. In our previous work [2], we developed an extension of this approach and presented dynamic replanning of footstep paths to be able to efficiently replan paths, thereby reusing information from previous searches. To obtain fast planning results for long distances, we previously relied on pre-computing footstep transitions across obstacle areas in order to estimate the traversal costs [1]. Afterwards, an adaptive planner switched between 2D path planning and footsteps depending on the difficulty of the area. However, pre-computation is not always feasible, and a densely cluttered environment would require estimating all possible paths through it.



Parameter	Value / Range
w	10 cm
h	16 cm
Δx	$[-10, 22]$ cm
Δy	$[12, 28]$ cm
$\Delta\theta$	$[-0.23, 40.0]°$

Fig. 2. Footstep transition model (left) and parameterization for a large humanoid such as HRP-2 or ASIMO, consisting of 14 different steps (middle and right).

Vernaza *et al.* [15] presented search-based planning for a quadruped robot. The authors compute global plans in a stance graph with R*. Compared to humanoid footstep planning, their quadruped only has point feet, and planning was performed in a relatively small area of rough terrain.

To the best of our knowledge, we present the first thorough evaluation of anytime search-based footstep planning with the aim of near-realtime plan results over long distances.

III. Planning Framework

A. States, Actions, Transition Model, and Lattice Graph

The robot's state is described by the global position and orientation of its stance foot $s = (x, y, \theta)$, alternating between the right and left foot.

A single footstep action is parameterized by the displacement of the moving foot (left or right) relative to the stance foot, as illustrated in Fig. 2. Under the constraint that humanoids move the left and the right leg alternating and that the possible footsteps for both feet are symmetric, a footstep action a can be parameterized by $a = (\Delta x, \Delta y, \Delta\theta)$ relative to the position of the stance foot.

When the footstep planner expands a state s, it determines all successor states s' resulting from applying the transitions of all available actions $a \in A$: $s' = t(s, a)$. The resulting states s' are checked for collisions with obstacles and invalid states are discarded.

The costs of a state transition from s to s' are given by the transition costs $c(s, s')$ that correspond to the execution time according to

$$c(s, s') = \|(x, y), (x', y')\| + k \tag{1}$$

for $s = (x, y, \theta)$ and $s' = (x', y', \theta')$. Here, k are constant costs associated to executing one footstep, leading to a penalization of paths with a higher number of steps. We hereby assume that the effort of changing the orientation of a footstep can be neglected compared to the Euclidean distance.

The iterative construction of states (x, y, θ) connected by footsteps builds a sparsely-connected lattice graph, where the single footstep actions correspond to the motion primitives used in search-based planning. In contrast to search-based motion planning for wheeled robots (e.g., [16], [17]), for humanoids the available motion primitives are alternated between left and right leg. In case only shallow obstacles are considered, solely the end foot location of a footstep action needs to be collision-free, as humanoids can step over these obstacles.

The footstep set for a large humanoid robot that we use throughout this work is shown in Fig. 2 (middle). Since this represents a discrete set of stepping motions derived from the kinematic range of the humanoid, the goal location (given as a pair of footsteps or a single state) may not be exactly reached, or require too many intermediate steps to reach. Thus, we dynamically extend the footstep set with a transition to the goal if it is reachable from the current state.

B. Environment Model and Collision Checking

We use a 2D grid map consisting of equally-sized grid cells that are marked as either free or occupied. As in [1], occupied cells also contain traversability information, enabling the robot to differentiate between shallow obstacles that can be stepped over (such as uneven floor or clutter), and obstacles that have to be avoided with a larger clearance (such as walls or furniture).

In order to validate a possible footstep, a 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. Here, we apply an efficient recursive collision check in the distance map [2] and assume that the humanoid's footprint is rectangular, or can be approximated by a rectangular bounding box.

C. Search-based Planning

On the lattice graph, search methods such as A* can be applied. The costs to transition between states are given from the transition model (1) and only valid states remain after collision checking. The search is guided towards the goal by a heuristic that can include different information.

Common heuristics for footstep planning are the straight-line Euclidean distance to the goal, or the estimated costs along a 2D path planned with A* or Dijkstra in a grid. The latter is potentially inadmissible because it does not reflect the humanoid's capability of stepping over obstacles [2], [3].

IV. Weighted A* Search

As a basis for both ARA* and R*, we will first recapitulate weighted A* (wA*) search. A* search expands states according to the evaluation function

$$f(s) = g(s) + h(s), \tag{2}$$

where $g(s)$ are the actual costs of the best path from the start to the current state s and the heuristic $h(s)$ provides the estimated costs to the goal from state s. $h(s, s')$ denotes the heuristic costs from a state s to s'. For the optimality of A* to hold, $h(s)$ must be admissible, i.e.

$$\forall s : h(s) \leq c(s, \text{goal}), \tag{3}$$

where $c(s, \text{goal})$ denotes the true costs of traversing from state s to the goal.

wA* inflates h with a factor $w \geq 1$ which results in suboptimal paths that can be found faster, thereby expanding fewer states. With this inflated heuristic, the quality of the solution can be traded off for efficiency, while the resulting paths are guaranteed to cost no more than w times the cost of an optimal path [7]. For $w = 1$, this corresponds to a regular A* search. As illustration, Fig. 3 shows the resulting footstep plans around an obstacle and the expanded states for different weights w.

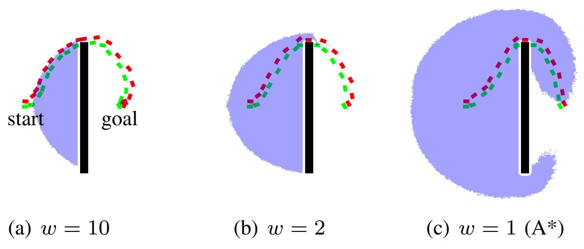


Fig. 3. wA* and ARA* footstep planning (the paths and expansions are identical) around an obstacle for different heuristic inflation weights w. Here, the Euclidean distance to the goal was used as heuristic. The expanded state area is shaded in blue. An inflated heuristic results in fewer expanded states and faster planning times at the cost of suboptimal paths.

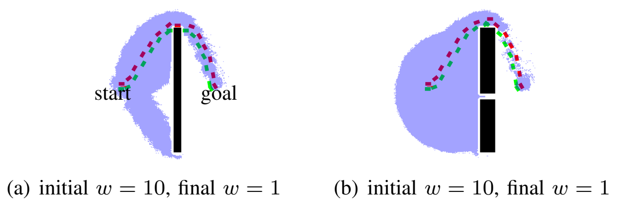

start
goal
(a) initial $w = 10$, final $w = 1$
(b) initial $w = 10$, final $w = 1$

Fig. 4. ARA* planning around a local minimum with a 2D Dijkstra path as informed heuristic. In (a), planning succeeded after 1.8 s with 267 288 expanded states. However, the heuristic wrongly leads into a non-traversable narrow passage in (b) where planning succeeded after 15.0 s with 1 892 608 expanded states.

V. Footstep Planning with ARA*

Anytime Repairing A* (ARA*) search runs a series of wA* searches while efficiently reusing previous information [7]. An initially large w causes the search to find a non-optimal initial solution quickly. Then, as time allows, the search reruns with incrementally lower values for w while reusing much of the information from previous iterations. Given enough time, ARA* finally searches with $w = 1$, producing an optimal path. If the planner runs out of time before, the cost of the best solution found is guaranteed to be no worse than w times the optimal solution cost. This allows ARA* to find some solution faster than regular A*, and to approach optimality as time allows.

As any heuristic search, the efficiency of ARA* depends on the quality of the used heuristic. The more informed it is of the environment, the fewer states need to be expanded. The standard heuristic of the Euclidean distance to the goal will create local minima around obstacles that block the straight path to the goal, and forces the planner to expand large areas of the state space (Fig. 3(c)).

Compared to that, using a 2D Dijkstra path to the goal as heuristic is better informed of obstacles. This results in expanding states on paths around the obstacles instead of local minima (Fig. 4(a)), and it has been shown to result in more efficient planning times [2], [3]. However, this heuristic

Algorithm 1: Single iteration of R*

```
select unexpanded state $s \in \Gamma$ (priority to states not labeled as AVOID)
if path of edge $bp(s) \rightarrow s$ not computed yet then
    try to compute path $bp(s) \rightarrow s$ with wA*
    if failed then label $s$ as AVOID
    else
        update $g(s)$ based on cost of found path and $g(bp(s))$
        if $g(s) > w \cdot h(s_{start}, s)$ then label $s$ as AVOID;
else // expand $s$ by growing $\Gamma$
    let $SUCCS(s)$ be $k$ randomly chosen states at distance $\Delta$ from s
    if goal within distance $\Delta$ from $s$ then add goal to $SUCCS(s)$;
    foreach $s' \in SUCCS(s)$ do
        add $s'$ and edge $s \rightarrow s'$ to $\Gamma$
        $bp(s') := s$ // set backpointer to predecessor $s$
```

is potentially inadmissible and may even result in inefficient or no solutions at all in the case of footstep planning. It may also be susceptible to other types of local minima, again leading to inefficient planning. As an example, consider Fig. 4(b) with a narrow passage through the obstacle. The 2D Dijkstra heuristic will lead the planner to expand states through this passage even though it is not traversable with stepping motions. "Inflating" the obstacles by the robot's circumcircle for computing the 2D heuristic path would prevent this specific problem, but essentially degrades the planning process to planning footsteps around a 2D path, resulting in non-optimal paths since the robot can no longer step through clutter [1]. To keep this capability, only the foot incircle can be used as the maximum obstacle inflation radius for the heuristic.

In case a location is surrounded by planar obstacles or clutter, there does not even exist a valid Dijkstra path as heuristic. This is the case e.g. with shallow steps, where each edge must be treated as an obstacle.

All in all, ARA* is highly dependent on a well-designed heuristic function for efficient results. Designing this heuristic becomes a challenging problem in itself for complex environments.

VI. Footstep Planning with R*

The randomized A* search (R*) aims to solve that problem by depending less on the quality of the heuristic function [8]. The search avoids local minima by running a series of short-range, fast wA* searches towards randomly chosen sub-goals. The exploration of the search space with random sub-goals relates to randomized planners such as RRTs. But contrary to them, R* aims to minimize the solution cost and provides probabilistic guarantees of the solution suboptimality [8]. In the context of navigation planning, R* is able to solve more problems and provide better solutions than RRTs.

R* iteratively constructs a graph Γ of sparsely placed states in the same lattice graph as the dense search (see Algorithm 1). At each iteration, a state in Γ is expanded by generating k random successor states at a distance Δ. Any goal state within Δ is added to the successors of s as well. For footstep planning, there can be up to two valid goal states, corresponding to the left and the right foot.

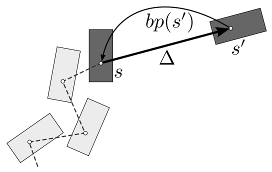
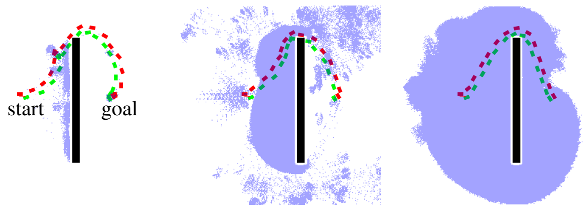


Fig. 5. R*-expansion of a state $s \in \Gamma$ (dark gray) to which a local footstep path (light gray / dashed) exists. k successors are determined in a random direction at distance Δ. If a successor s' is collision-free, it is added to Γ with the edge $s \rightarrow s'$ and the pointer to its predecessor $bp(s') = s$ is stored. At the next expansion of s', the search tries to find a local footstep path between s and s'.



(a) $w = 10$ (b) $w = 2$ (c) $w = 1$

Fig. 6. R* iterations for different heuristic inflation weights w. As heuristic the Euclidean distance to the goal was used.

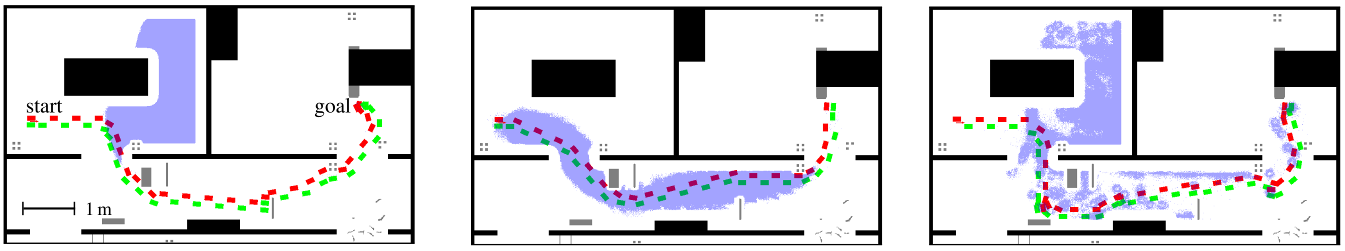
Each edge in Γ between two random states corresponds to a path in the original, dense search graph, which is determined by a local search with wA*. R* hereby first tries to find all "easy" local paths. If a local path requires too many expansions, it is labeled as AVOID and will only be planned when required to meet the suboptimality bounds. During the whole expansion from start to goal, R* provides probabilistic guarantees on the suboptimality of the solution by heuristic inflation w of the local wA* search. As in ARA*, R* iteratively lowers w and re-runs the search if a given planning time limit permits.

Generating the random nodes in Γ ensures the exploration of the search space. For footstep planning with humanoid robots, we randomly sample a direction and place a state s' at distance Δ from the current state, as illustrated in Fig. 5. The leg of the state (left or right) is also chosen randomly, whereas its orientation is given by the initial random direction to favor forward walking. If s' is collision-free, it is added to Γ along with the edge $s \rightarrow s'$.

Examples for different weights in the scenario with a local minimum are shown in Fig. 6, using the Euclidean distance heuristic. At the beginning, R* sparsely samples the state space towards the goal and around the obstacle. As the heuristic inflation w approaches 1, the state space expansion grows more dense in order to find the optimal path. Only then the expansion resembles wA* and ARA* expansions.

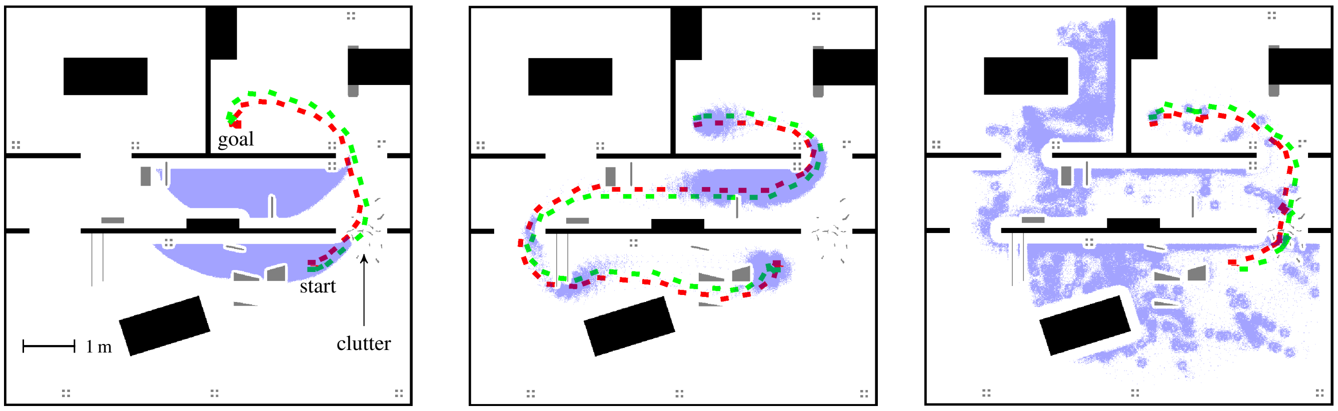
VII. Experiments

In the experiments, we evaluate and compare the performance of ARA* and R* for footstep planning. We chose a lattice graph resolution (discretization) of 1 cm for x/y and $5°$ for θ with the footstep parameterization from Fig. 2. This discretization is used to check equality when expanding states on the lattice. The occupancy maps for collision checks have



(a) ARA* with Euclidean heuristic after 43 s (b) ARA* with Dijkstra heuristic after 5 s (c) R* with Euclidean heuristic after 5 s

Fig. 7. Footstep planning with a time limit of 5 seconds between two rooms of an indoor environment. Gray areas denote shallow obstacles that the robot can step over, black areas denote objects that the robot has to avoid (tables and walls). ARA* with the Euclidean heuristic fails to find a path within the time limit and requires 43 s to find a first solution with $w = 10$. With the Dijkstra heuristic, ARA* finds a near-optimal path within the time limit (final $w = 1.4$). R* finds a path even with the Euclidean heuristic (final $w = 7$).



(a) ARA* with Euclidean heuristic after 92 s (b) ARA* with Dijkstra heuristic after 5 s (c) R* with Euclidean heuristic after 5 s

Fig. 8. Footstep planning with a time limit of 5 seconds through a cluttered passage. ARA* with the Euclidean heuristic fails to find a path within the time limit and requires 92 s to find a first solution with $w = 10$. The inadmissible Dijkstra heuristic results in a detour due to the clutter blocking the heuristic path. R* finds a path even with the Euclidean heuristic (final $w = 8$).

a resolution of 1 cm and angles are preserved to be continuous in the collision check. The robot needs to maintain a clearance of 15 cm to obstacles marked as walls. The inflation radius for the 2D Dijkstra heuristic is the foot incircle radius of 5 cm, as explained in Sec. V. We experimentally determined Δ=1.5 m, k=20, and the expansion limit for easy-to-find paths as 500 for R*. Planning a distance of 1.5 m in collision-free space requires 100–200 A* expansions with our footstep parameterization. All planning times are given for a single core of a desktop CPU (Intel Core i7, 3.4 GHz).

A. Planning in an Indoor Environment with Limited Time

In a first experiment, we evaluate the algorithms qualitatively in an indoor environment containing several rooms with shallow obstacles and obstacles that have to be avoided (walls). To obtain near-realtime plans suitable for navigation, we set the time limit for the planners to 5 seconds and the initial heuristic inflation weight w=10. The results are shown in Fig. 7. With the Euclidean distance heuristic, ARA* fails to find a plan within the given time. It requires 43 seconds for a first solution due to expansions being misled into local minima. In this planning scenario, the 2D Dijkstra path heuristic is very well suited due to the connectivity of the rooms through the hallway and relatively few planar obstacles. This leads ARA* with the Dijkstra heuristic to find a first solution after 0.7 ms and a near-optimal path (w=1.4) within 5 s. In comparison, R* with the Euclidean heuristic finds a first solution after 1 s and a solution with w=7 within the time limit of 5 s. It expands fewer states than ARA* with the same heuristic. Since we are mainly interested in using this admissible heuristic, we omit results of R* with the Dijkstra heuristic.

The problem of using the 2D Dijkstra heuristic becomes evident in a second scenario with different start and goal locations. Here, the optimal path leads through a passage blocked by clutter that the robot can step over (Fig. 8). A similar situation arises when there is a door sill that the robot should avoid stepping onto, which poses an obstacle completely blocking the doorway in 2D. Again, ARA* with the Euclidean distance heuristic fails to find a path within reasonable time while R* succeeded within the time limit of 5 seconds. In this scenario, the Dijkstra heuristic is no longer admissible since the clutter blocks its path through the doorway. As a result, ARA* with this heuristic wrongly expands a non-optimal path despite w reaching 1.4 after 5 s. This is because the suboptimality guarantees no longer hold due to the heuristic inadmissibility, and demonstrates that the Dijkstra heuristic is a poor choice for footstep planning in cluttered environments.

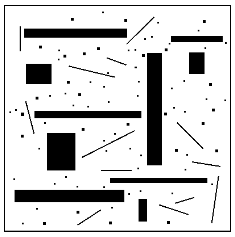
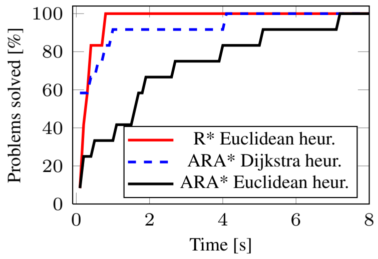



Fig. 9. A densely cluttered area (left) and the success rate for 12 random start and goal configurations in it (right). The percentage of solved problems within a certain time is shown for the first solution with w=5.

TABLE I
PERFORMANCE COMPARISON FOR FIRST SOLUTIONS IN A DENSELY CLUTTERED ENVIRONMENT

Planner	Heuristic	Planning time [s]	Path costs
R* (w=5)	Euclidean	0.32 ± 0.23	16.45 ± 3.16
ARA* (w=5)	Euclidean	2.15 ± 2.21	13.57 ± 1.15
ARA* (w=5)	2D Dijkstra	0.56 ± 1.13	20.41 ± 5.08
Optimal: A* (w=1)	Euclidean	33.31 ± 15.00	11.06 ± 1.20

B. Planning Through Dense Clutter: Initial Plan Results

The next set of experiments is designed to evaluate how the different planners can cope with densely cluttered scenes such as the $4 \times 4\,\mathrm{m}^2$ area shown in Fig. 9 (left). Here, we analyze the planning time and path quality of the first solution. To this end, we plan footstep paths between 12 random start and goal locations approximately 3.5 m apart. A* with the Euclidean distance heuristic requires on average 33 s to find the optimal paths containing 37 single steps on average. Fig. 9 (right) shows the success rate for the anytime planners to find the first solution with w=5 and Table I shows the aggregated statistics as mean and standard deviation.

As in the indoor experiments, R* succeeds in finding fast results in general. ARA* with the admissible Euclidean heuristic requires more planning time even for initial suboptimal results, while it needs less time with the inadmissible Dijkstra heuristic. The Dijkstra heuristic also leads to some longer paths, since it overestimates in some instances. All anytime planners find paths significantly faster than A*, at the cost of longer paths for the initial solution. Fig. 1 shows the paths and expanded states in comparison for one start and goal location. Given enough planning time, both R* and ARA* with the Euclidean heuristic will converge to the optimal result from A* planning.

With a localization system running to correct for motion drift, the planned paths can be directly used for humanoid navigation [2]. The robot can hereby continue to plan while executing an already planned suboptimal solution, dynamically switching to the better plan after each step.

VIII. Conclusions and Future Work

In this paper, we presented footstep planning with the anytime search algorithms ARA* and R*. As we demonstrated in the experiments, this enables planning efficient paths with provable suboptimality within short planning times. We found ARA* with the 2D Dijkstra path heuristic suitable for some planning scenarios, but generally in danger of yielding non-optimal paths. ARA* with the admissible Euclidean heuristic takes too much time and expands too many states in the presence of local minima. R*, on the other hand, yields fast solutions even with the Euclidean heuristic and avoids local minima.

Although plan execution and re-planning was not the focus of this work, an anytime variant of D* can be employed similar as in our previous work [1], [2]. This enables efficient plan re-usage in case of updated localization estimates or changes in the environment while the robot is walking. Our approach can be extended to 3D when collision checks are also performed between the robot's 3D model and the environment, similar to [5].

References

[1] A. Hornung and M. Bennewitz, "Adaptive level-of-detail planning for efficient humanoid navigation," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.
[2] J. Garimort, A. Hornung, and M. Bennewitz, "Humanoid navigation with dynamic footstep plans," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.
[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] 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*, vol. 28, pp. 427–439, 2012.
[6] 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.
[7] 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.
[8] M. Likhachev and A. Stentz, "R* search," in *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2008.
[9] M. Likhachev, http://www.ros.org/wiki/sbpl, 2010.
[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] 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.
[13] 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.
[14] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.
[15] P. Vernaza, M. Likhachev, S. Bhattacharya, S. Chitta, A. Kushleyev, and D. D. Lee, "Search-based planning for a legged robot over rough terrain," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.
[16] M. Likhachev and D. Ferguson, "Planning long dynamically-feasible maneuvers for autonomous vehicles," in *Int. Journal of Robotics Research (IJRR)*, 2009.
[17] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.