Real-Time Path Planning in Unknown Environments for Bipedal Robots

Arne-Christoph Hildebrandt, Moritz Klischat, Daniel Wahrmann, Robert Wittmann, Felix Sygulla, Philipp Seiwald, Daniel Rixen and Thomas Buschmann[1]

***Abstract*—Autonomous navigation in dynamic and unknown environments requires real-time path planning. Solving the path planning problem for bipedal locomotion quickly and robustly is one of the main challenges in making humanoid robots competitive against mobile platforms. In this paper, we propose strategies to use mobile platform planners for improving the navigation of bipedal robots. These strategies combine advantageously continuous 2D paths with conventional step planners for humanoid robots. We introduce a mobile platform planner suitable for real-time navigation. It searches for multiple 2D paths which makes the path planning more robust against limited calculation time and changing scenarios. It is combined with a step planner and integrated in the framework for autonomous navigation of our robot *Lola*. We evaluate different strategies in simulation and validate them in experiments in unknown dynamic environments.**

***Index Terms*—Humanoid and Bipedal Locomotion, Reactive and Sensor-Based Planning, Motion and Path Planning, Visual-Based Navigation**

I. Introduction

LEGGED robots are suited to be employed in diverse and complex scenarios. Due to their kinematic capabilities, they can outperform wheeled robots in cluttered environments. We expect robots to be able to react to dynamically changing real-world scenarios. Therefore, motion planning for legged robots has to solve the high dimensional motion problem quickly and reliably. Most current humanoid control frameworks follow a hierarchical approach, which allows, up to a certain level, to separate the navigation problem in complex environments from the walking pattern generation to achieve real-time bipedal walking [1]–[3]. The navigation problem can then be reduced to the search of consecutive footholds which are executed by the robot. In contrast to a continuous 2D path planning problem (e.g. cars) path planning methods for biped locomotion have to consider the partially discrete character of bipedal locomotion. On the one hand, the discrete character of the consecutive footholds complicates the motion planning problem, since the search space is largely augmented. On the other hand, consequently, it gives the biped robot more possibilities to navigate in complex environments, for example by traversing obstacles. In environments that include non-traversable obstacles, the search for discrete consecutive footholds gets complex and time-consuming ([4], [5]). Therefore, it seems to be reasonable to consider the environment at different detail levels in order to improve the performance of the step planner.

In this paper, we introduce a mobile platform planner which is used for *2D Pre-Planning* in a reduced environment representation to accelerate a more detailed step planner. In contrast to commonly used mobile platform planners, our does not only provide one 2D path, but multiple 2D paths. This is especially important for exploiting the capacities of bipedal walking in cluttered environments.

Furthermore, we focus on the combination of the mobile platform planner and the step planner. Different possibilities are proposed and evaluated. These coupling methods can be applied in conjunction with any 2D path to accelerate step planning.

Using the proposed method, the robot's reactivity in dynamically changing environments is significantly improved.

The paper is organized as follows: In Section II, we present related work. Section III provides an overview of the experimental platform used in this work – the robot *Lola* – and its framework for real-time motion generation. In Section IV, we then present our approach for *2D Pre-Planning* for legged robots. The method is analyzed in simulation and validated in successfully conducted experiments. The results are presented in Section V. Finally, Section VI is devoted to the conclusion and comments on future work.

II. Literature Review

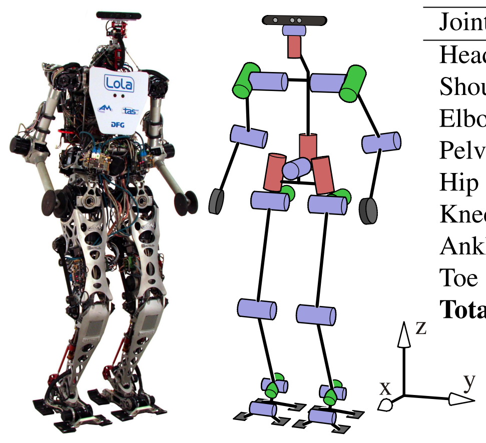
In humanoid robotics, the path planning problem is often solved by first discretizing the set of the robot's kinematically reachable footholds (*action set*). Based on this fixed set of discrete footholds, implicit graph search algorithms like A*–Search (e.g. [2]) are applied to find an executable step sequence. In real-world scenarios with dynamically changing environments computation time plays a crucial role. The heuristics used in search algorithms to guide the search to the goal are major keys to reduce planning time. In [6] the authors evaluate the influence of the heuristics to improve the speed of an A*–Search based step planner. They propose to dynamically adapt the weighting factor of the heuristic based on the Euclidean distance to the goal. Nevertheless, a poorly chosen heuristic could still lead the search in wrong directions. In [5],

Manuscript received: February, 15, 2017; Revised April, 28, 2017; Accepted May, 28, 2017.

This paper was recommended for publication by Editor Nikos Tsagarakis upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by the DAAD and the *Deutsche Forschungsgemeinschaft* (BU 2736/1-1).

[1]Chair of Applied Mechanics, Technical University of Munich, 85747 Garching, GER E-Mail: Email: {arne.hildebrandt, rixen}@tum.de Phone: +49 89289 15208

Digital Object Identifier (DOI): see top of this page.



Joint	DoF
Head	2
Shoulder	2
Elbow	1
Pelvis	2
Hip	3
Knee	1
Ankle	2
Toe	1
Total	**24**

Fig. 1: Photo and kinematic structure of the humanoid robot *Lola*. Joint distribution and used world coordinate system are shown on the right side [4].

[7], [8] hierarchical path planning methods are proposed. Their common idea is to combine a simplified global mobile platform planner with a detailed local step planner. The mobile platform planner calculates a 2D path in an simplified environment representation which is used to guide the local step planner search. [5] presents an hierarchical path planning approach for branched buildings, which uses three levels of detail: On the top-level, a global 2D path is divided into sub-goals. An A*-based mobile platform planner searches backwards from the sub-goal to provide the low-level step planner a heuristic. [7] proposes to first plan a global 2D path using an A*–Search, as well. Then sub-goals are generated from the 2D path. The local step planner first constructs a path from line segments which is subsequently used for geometrically generating the foothold positions. This allows for real-time capability. However, the robot's kinematic capacities such as stepping over obstacles are not fully exploited, nor is the solution's optimality explicitly considered. [8] proposes to use a combination of a mobile platform planner and a detailed step planner. In contrast to previous authors, they propose to switch between a mobile platform planner in regions without obstacles and to use a more sophisticated step planner in cluttered regions which allows for stepping over obstacles. This method's performance relies heavily on the availability of obstacle-free space. [9] uses a simplified approach designed for finding multiple paths which resembles the principle of visibility graphs. Sub-goals are placed next to the edges of obstacles and connected by straight step sequences. Since a relatively simple step planner is used, the resultant paths are not optimal. They are restricted to rather simple environments, even though small obstacles can be traversed. Unlike the work described above, the authors of [10] do not use a global map, but they propose to directly test a set of 2D tentacles on image data from a stereo vision camera whether or not they are viable. These pre-defined tentacles do not take a specific goal into account. This enables safe and reactive robot movements in an unknown environment using only on-board sensing. This system, however, does not allow the robot to step onto or over objects. Furthermore, [5] discusses the benefit of a heuristic provided by a mobile platform planner. It is shown that the use of that heuristic, which is adapted to the environment, helps to significantly accelerate the A*-based step planner. Nevertheless, when using the mobile platform planner, the ability of the robot to step over obstacles or onto obstacles is neglected in [5]. Similar to the cited publications, we want to exploit the possibility to accelerate the step planner by providing a continuous guideline calculated by a mobile platform planner. However, our approach differs from the previous ones:

(1) different levels of environment details are used for both planners which exploits the kinematic capabilities of bipedal robots (see Subsection IV-A).

(2) we do not use sub-goals as presented in [5], [7], [9]. Such sub-goals are intermediate targets that can be used to guide the A*–search by including the Euclidian distance to those sub-goals in the cost evaluation. In contrast, we propose and analyze different strategies to integrate a continuous 2D path in the A*–Search. Nevertheless, sub-goals do not contradict our method, but could easily be integrated in our approach. This would be particularly beneficial in large and complex environments like branched buildings, as an additional level-of-detail. Note however that, sub-goals guide the search only via heuristics. A continuous 2D path allows for further coupling methods, which will be discussed in the following sections.

(3) in contrast to [5], [7], [8] we do not search for one final 2D path, but for different candidates instead. The set of 2D paths guides multiple searches of the detailed step planner. This approach is necessary as the mobile platform planner only uses a reduced map. Therefore, the quality of the provided 2D paths greatly depends on the presence of traversable obstacles or stairs not considered in the pre-planning. This approach is mainly suitable for real-time applications such as navigation in unknown areas - the robot has to react quickly to the newly discovered environment without global knowledge about the environment (in contrast to [7], [9]). The effectiveness of our methods are moreover validated on a real humanoid robot.

III. Hardware and Control Architecture

We validate our strategies on our humanoid robot *Lola* (see Fig. 1). In the following, we present the hardware and the integration of our navigation module in the overall control architecture. In [1] more details on the mechanical design and the control architecture are given. Furthermore, we give an insight into the previous implementation of our step planner [4].

A. Hardware

Lola weights approximately 60 kg and is 180 cm tall. It has 24 position-controlled joints. The redundant kinematic configuration (see right side of Fig. 1) gives *Lola* a large action radius and allows the execution of complex motions. For environment recognition we use an Asus Xtion PRO LIVE RGB-D camera[1]. The vision processing software runs on an

[1]ASUS Xtion PRO LIVE, see http://www.asus.com/Multimedia/Xtion_PRO_LIVE/

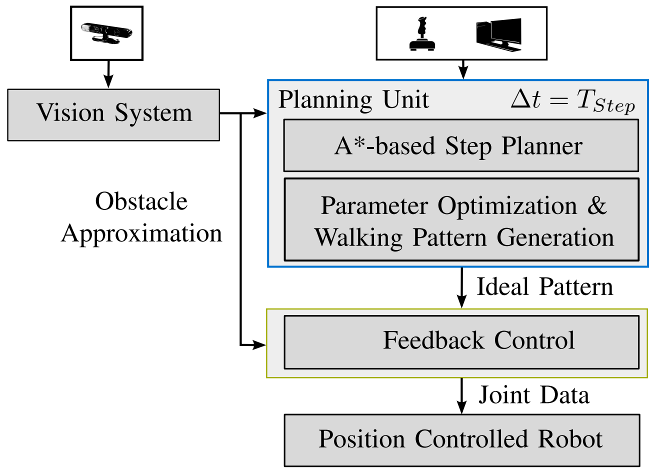

Vision System
Obstacle Approximation
Planning Unit $\Delta t = T_{Step}$
A*-based Step Planner
Parameter Optimization & Walking Pattern Generation
Ideal Pattern
Feedback Control
Joint Data
Position Controlled Robot

Fig. 2: *Lola*'s real-time walking control system.

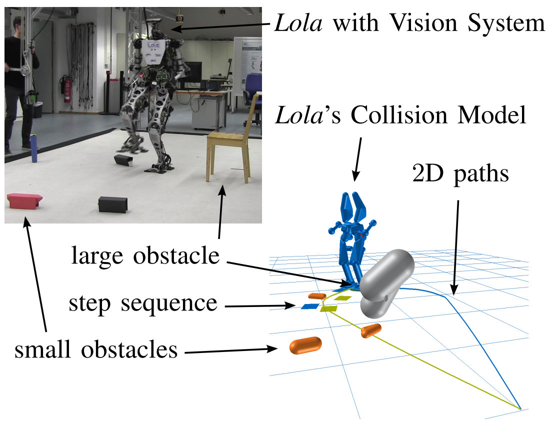


Fig. 3: Real-world scenario: static environment with small and large obstacles, *Lola* with on-board vision system, SSV approximation of *Lola* and of obstacles currently in the camera's field of view, calculated 2D paths and calculated step sequence.

on-board computer with an Intel Core i7-4770S @ 3.1 GHz (4x) processor and 8GB RAM on a Linux OS. The control software runs on a computer with the same specification, but on a QNX-RTOS. Both computers communicate via Ethernet using TCP.

B. *Control Architecture*

Our walking control system as depicted in Fig. 2 follows a hierarchical approach. The *Vision System* approximates the environment with swept-sphere volumes (SSV), which are used throughout our control approach for distance calculations. It only uses an on-board RGB-D sensor which works with a sampling time of 30 ms. This makes it possible to track moving obstacles while the robot is walking [11]. The navigation module [4], labeled as *A*-based Step Planner*, is part of the planning unit, which is called once every walking step. The step time T_{Step} varies between 0.6 and 1.2s. It gets desired step parameters, like the step length, desired goal positions, or a desired velocity vector, from the user input. Due to a cycle time of T_{step} it is very reactive to changes in the environment or in the user's input. Based on this information the navigation module calculates a sequence of parameter sets describing the walking pattern. Using the parameter set as an initial solution, the *Parameter Optimization & Walking Pattern Generation*-module ([12]) optimizes the parameter set and calculates kinematically feasible, collision-free, and dynamically executable trajectories. These trajectories are adapted according to sensor feedback in the *Feedback Control* unit with a cycle time of 1 ms and executed by the robot.

C. *A*–based Step Planner*

Step sequences are calculated by an A*–search based step planner [4]. Based on a discretization of the robot's kinematically reachable footholds, a search tree is implicitly constructed. An A*–search is applied to search for the optimal step sequence in terms of path costs.
In each search tree expansion the set of foothold suggestions is checked for their viability. Instead of checking in a binary way on a grid based map for collisions, we use our SSV-representations for collisions checks. That way, we can analyze the step suggestions in 3D applying a robot model which approximates the kinematic movement of the robot.
Furthermore, the set of foothold suggestions is rated by costs c_t. It includes traveled path length c_p, costs of the current step c_s and remaining path length to the goal estimated by a heuristic c_h:

$$c_t = c_p + c_s + c_h \tag{1}$$

The subsequent foothold to be expanded is then chosen by the lowest costs c_t of all steps which have not been analyzed yet. By including a goal heuristic, the A*–search is directed towards the goal. So far, the Euclidean distance was used as a heuristic. Since each step is linked to its predecessor, a valid step sequence can be retrieved at any time by tracing the predecessors of the currently best-rated step suggestion. This makes it suitable for real-time application. The search tree is constructed until the goal is reached, a desired number of steps is found, or a time limit is reached. [2]

IV. Proposed Method

Step planners as the one described in Subsection III-C have performance limitations. The goal heuristic based on the Euclidean distance helps to accelerate the search in the goal direction. If the path following the Euclidean distance is non-traversable (e.g. it is blocked by large obstacles or non-steppable platforms) the search is slowed down. The major reason for the increasing calculation time is the Euclidean heuristic, which does not consider the robot's environment. As a consequence, the remaining goal distance may be greatly underestimated. The search tends to explore irrelevant areas and a large number of irrelevant step suggestions are evaluated. This requires a lengthy computational time and the step planner

[2]A video showing public experiments with the current step planner in different environments, inluding platforms, and the performance of the overall system is available under https://youtu.be/g6UACMHgt20.

becomes impracticable for real-time applications.
We introduce an additional pre-planning procedure before the detailed step planner. The *2D Pre-Planning* searches for a continuous 2D path. Considering the environment, the 2D path represents the remaining distance to the goal more realistically than the uninformed Euclidean distance.

A. 2D Pre-Planning

Mobile platform planners plan continuous paths and, therefore, do not take into account the robot's capability to step over obstacles. The bipedal robot could traverse obstacles which the continuous mobile platform planner would avoid. Following a continuous path with a bipedal robot could unnecessarily lead the robot walk a long way around. The concept of the *2D Pre-Planning* is summarized in Algorithm 1.

Algorithm 1 *2D Pre-Planning*

```
 1: procedure 2D Pre-Planning($ReducedMap$, $Goal$, $n$)
 2:     for $i < n$ do
 3:         $path(i) \leftarrow$ InitialPath($i$, $Goal$)
 4:         while $collision = true$ do
 5:             $path(i) \leftarrow$ PotentialAdaption($path(i)$)
 6:             $collision \leftarrow$ CheckCollision($path(i)$)
 7:     $SelectedPaths \leftarrow$ SeperatePaths($path$)
 8:     for $i < n$ do
 9:         if $SelectedPaths(i) = true$ then
10:             $path(i) \leftarrow$ ElasticBandOpt($path(i)$)
```

Environment Modeling: In order to preserve the robot's ability to traverse obstacles, only non-traversable obstacles are taken into account during *2D Pre-Planning*. Since obstacles are clustered and approximated by combinations of SSV objects, this is efficiently implemented by checking for the obstacle's dimensions. In the subsequent detailed step planning, the full map is used in turn (see Fig. 4).

Robot Modeling: We simplify the robot's approximation for *2D Pre-Planning*. The mobile platform planner searches for continuous paths. Therefore, the robot is represented by a point model with an additional safety margin. This further simplifies collision checking and detection.

Reduced Map: Standard path planning algorithms for mobile platforms ([13], [14]) search for one continuous 2D path. The reduced environment map, used by the *2D Pre-Planning*,

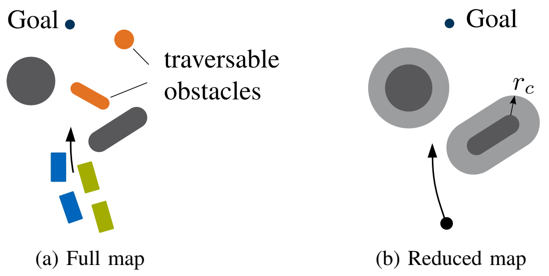


Fig. 4: Reduction of the environment map.

induces an uncertainty regarding the resultant costs of the subsequent step sequence. The optimal path which is found by the mobile platform planner could turn out to be sub-optimal when planning the step sequence and using the full map (see Fig. 5). In the presence of multiple small objects on the 2D–path, the optimal path could even not be traversable. For this reason, the proposed mobile platform planner is designed to find various path variants.

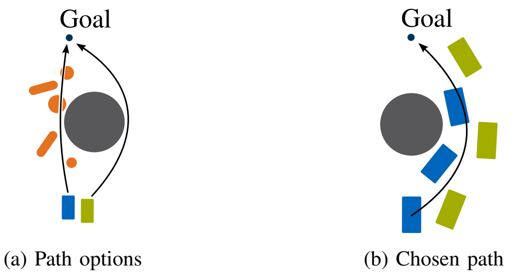


Fig. 5: Several continuous 2D–paths in presence of obstacles. Grey obstacle is non-traversable by bipedal robot. Orange obstacles are traversable.

Initial Paths: The initial paths are generated by a set of n_{init} parabolas which connect start and goal position (see Fig. 6). The parabolas are parameterized with the parameter s which defines the distance perpendicular to the direct connection between start and goal position. The direction is defined by the unit vector $\boldsymbol{n}$. n_{init} and s are predefined by the user and determine the discretization of the area.[3] Using parabolas at this stage is not paramount for the success of the procedure. The key point is that we use curves which covers the area in which the robot has to navigate with a set of initial and different solutions. Each parabola g is discretized with $n_{S,g}$ supporting points $\boldsymbol{x}_{s,g,i}$ lying on the initial parabolas. The supporting points are connected via splines.

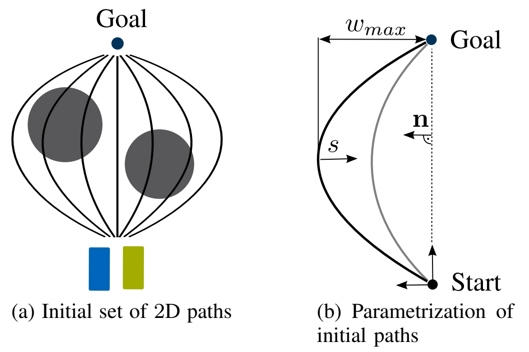

Goal
(a) Initial set of 2D paths
w_{max}
Goal
$\mathbf{n}$
s
Start
(b) Parametrization of initial paths

Fig. 6: Discretization - Generating initial solutions.

Collision-free Paths: Up to this point, $\boldsymbol{x}_{s,g,i}$ are created based on the set of parabolas without knowledge of the obstacles. In the next step, $\boldsymbol{x}_{s,g,i}$ which lie in obstacles are shifted iteratively to become collision-free. For generating collision-free paths, we use a potential approach widely applied in path planning [15]. We present this approach for one exemplary path, but the method is used simultaneously for all paths

[3]In this work, vectors are represented as $\boldsymbol{a} \in \mathbb{R}^2$ and scalars as a.

generated from the initial set of parabolas. Each obstacle j is modeled by an artificial potential ϕ_j as

$$\phi_j(\boldsymbol{x}^k_{s,g,i}) = \ln\left(r_j(\boldsymbol{x}^k_{s,g,i})\right) \tag{2}$$

with the shortest distance $r_j(\boldsymbol{x}^k_{s,g,i})$ between an obstacle j and a given point $\boldsymbol{x}^k_{s,g,i}$ in iteration step k. Note: We use the same library for distance calculation as used in our methods for collision avoidance [16]. Superposition of source terms of all n_{obs} obstacles results in the potential equation

$$\phi(\boldsymbol{x}^k_{s,g,i}) = \sum_{j=1}^{n_{obs}} \ln\left(r_j(\boldsymbol{x}^k_{s,g,i})\right) \tag{3}$$

For shifting each supporting point i, we calculate numerically the derivative $\nabla_{\boldsymbol{n}}\phi(\boldsymbol{x}^k_{s,g,i})$ with respect to $\boldsymbol{n}$ using the perturbation ε:

$$\nabla_{\boldsymbol{n}}\phi(\boldsymbol{x}^k_{s,g,i}) = \frac{\phi(\boldsymbol{x}^k_{s,g,i} + \varepsilon \cdot \boldsymbol{n}) - \phi(\boldsymbol{x}^k_{s,g,i})}{\varepsilon} \tag{4}$$

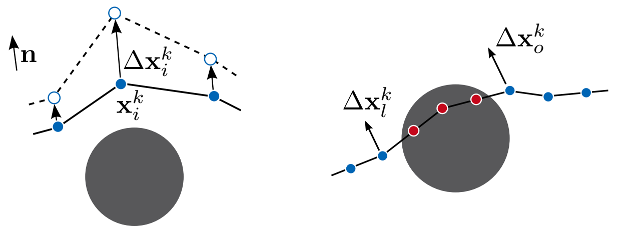
The translation increment is computed using a scaling parameter c_{trans}

$$\Delta\boldsymbol{x}^k_{s,g,i} = \boldsymbol{n} \cdot c_{trans} \cdot \nabla_{\boldsymbol{n}}\phi(\boldsymbol{x}^k_{s,g,i}). \tag{5}$$

$\boldsymbol{x}_{s,g,i}$ is updated as follows (see Fig. 7 (a))

$$\boldsymbol{x}^{k+1}_{s,g,i} = \boldsymbol{x}^k_{s,g,i} + \Delta\boldsymbol{x}^k_{s,g,i}. \tag{6}$$

If a supporting point is located inside an obstacle, no potential is defined, since obstacles may consist of more than one SSV. In this case, the increment $\Delta\boldsymbol{x}^k_{s,g,i}$ is set to the maximal $\Delta\boldsymbol{x}_{s,g,j}$ in direction $\boldsymbol{n}$ from both neighboring supporting points l and o which are still collision-free (see Fig. 7 (b)). The iterative translation is terminated when either all paths are collision-free or a maximal number of iterations has been reached.



(a) Outside obstacles (b) Inside obstacles

Fig. 7: Generating collision-free paths – updating of supporting points.

Separating Paths: In a subsequent step, we optimize the paths to obtain not only collision-free, but optimal paths with respect to the path length. To reduce computational effort, we only consider collision-free paths which tend to converge to different final paths. This happens when there is at least one obstacle between two paths (see Fig. 5). We first identify each subset of collision-free paths which are not separated by an obstacle. Then we select the shortest path of each subset for further optimization.

Optimizing Paths: In order to ensure optimality and smooth paths, the selected collision-free paths are optimized seperately. We use the *elastic band method*, which simulates a contracting force acting in an elastic band. It is described in [17].

B. Coupling with Step Planner

The 2D paths are passed to the A*-based step planner (compare Subsection III-C). The methods presented here are applied to our A*-based step planner, but they can be combined with any node-based search algorithm for bipedal locomotion. Overall, we propose three different methods to use the 2D path for accelerating the search. These can be applied independently from each other as well as in combination:

Heuristic: In this approach the Euclidean distance to the goal used in the cost evaluation is replaced by the distance along the 2D path. That way, the heuristic can better estimate the remaining path costs. This accelerates the A*–search significantly (see [5]). The heuristic costs c_h are computed by orthogonally projecting the position $\boldsymbol{x}_{sug}$ of a step suggestion onto the 2D path. Integrating the arc length from the projection $\boldsymbol{x}_{sug,\perp}$ to the goal yields the remaining path length l_g. To prevent the A*–search from deviating too much from the 2D path, the lateral distance from the path weighted by a factor $w_\perp$ is incorporated in the heuristic as well:

$$c_h = l_g + w_\perp \|\boldsymbol{x}_{sug} - \boldsymbol{x}_{sug,\perp}\| \tag{7}$$

For computing the projection and the remaining length a linear spline representation is used for connecting the supporting points and approximating the 2D path.

Restriction of Search Area: It is assumed that the optimal step sequence is located within the immediate vicinity of the 2D path. Therefore, the search area S of the A*–search can be restricted to a maximal distance d_{max} from the 2D path. All step suggestions which lie outside the search area are omitted, since they are considered to lead to sub-optimal step sequences:

$$\|\boldsymbol{x}_{sug} - \boldsymbol{x}_{sug,\perp}\| < d_{max}. \tag{8}$$

Consequently, the number of investigated step suggestions will be reduced. Choosing a sufficiently large value d_{max} still allows the step planner to react to small obstacles which were ignored during *2D Pre-Planning*.

Reduction of Search Space Dimension: Due to the tree structure of the A*–search graph, a reduction of the search space dimension tends to decrease the number of investigated step suggestions exponentially. As described in [4], the state of each node consists of the relative displacement $\boldsymbol{x}_{sug}$ and the orientation φ of each foot. By orienting the angle φ relative to the 2D path we no longer consider φ in the search space S of the A*–search. That way, the search space is reduced by one dimension. This greatly accelerates the search. φ is computed as the tangent's angle φ_t of the projection $\boldsymbol{x}_{sug,\perp}$ on the 2D path of $\boldsymbol{x}_{sug}$.

C. Real-Time Implementation

The proposed methods are implemented for real-time applications. Fig. 3 depicts a real-world scenario with small and large obstacles, the approximations of *Lola* and of the obstacles as SSV elements, calculated 2D paths and the step sequence. To ensure reactive planning to environmental changes or changes in the user input, the map is updated after each physical step of the robot. Both *2D Pre-Planning* and step planning replan before executing the next step. Due to the real-time constraints, the step planning is implemented such, that it can be aborted at any given time. The resulting step sequence would be sub-optimal, but executable by the robot. Our vision system, as most on-board vision systems, has a restricted field of view. Furthermore, the level-of-detail of our environment approximation is distance-dependent to account for sensor noise and to reduce computational efforts [11]. Therefore, in experiments, planning for the entire path to the goal is hardly practical and is not necessarily advantageous. Therefore the planning is only performed for a predetermined number of steps: we abort the search after $n_{exp,des}$ steps are found [4]. If no feasible step sequence can be found during the set time limit (approx. 400ms) and using the shortest 2D path, the step planner searches for a valid sequence to walk on the spot. Since we obtain multiple 2D paths after *2D Pre-Planning* the step planner could ideally be parallelized on a multi-core processor in order to compare step sequences for different 2D paths. However, we have not parallelized the calculations yet, since it has not been necessary so far in our experiments.

V. Results

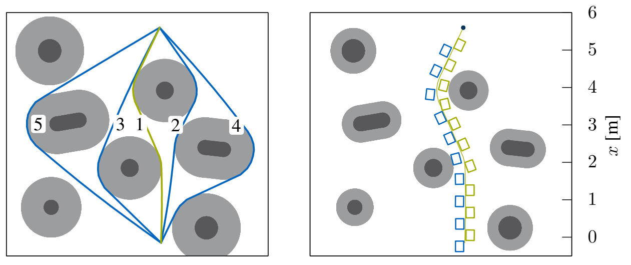
We evaluated the proposed methods in simulation [1] and validated the real-time character in experiments.

A. Simulation

In the following we compare the step planner with and without *2D Pre-Planning* and the different methods presented in Subsection IV-B. The test cases presented here are chosen to discuss and illustrate different aspects of our methods. We show longer step sequences than in real experiments, which may result in uncommonly-long calculation times. The first test case represents a simple environment similar to Fig. 9. The results are summarized in Tab. I. Using the method denomination mentioned in the legend of Tab. I, we can state the following: When using only (none) or (2) the path planning was not able to find a result in the set time limit. The results for the combination of (1) and (2) are nearly the same as for (1) alone, since the heuristic already guides the search close to the 2D path in this particular example. Results for (2) combined with (3) range in between those for (3) alone and all methods activated. Step planning with all methods combined yields the shortest search time even though the costs of the resulting step sequence are slightly higher. This is mainly due to the combination of (1) and (3) as confirmed by the result of this combination. All analyzed environments show similar results. For navigation in unknown environment we accepted slightly raised costs in favor of a significantly reduced search time. Therefore in the following all methods are activated.

TABLE I: Results for different methods: (none) Euclidean distance as heuristic, (1) 2D path heuristic, (2) restriction of search area, (3) reduced search space, (1,2,3) all methods combined.

Coupling methods	none	1	2	3	1,2,3
goal reached in time	no	yes	no	yes	yes
2D–path length [m]	-	5.62	5.62	5.62	5.62
costs	-	5.990	-	7.151	6.900
number of steps	12	14	14	18	18
distance calculations	324428	403710	401942	131446	13596
search time [s]	60.016	16.607	60.001	7.151	0.169
		1 & 2	1 & 3	2 & 3	
goal reached in time		yes	yes	yes	
2D–path length [m]		5.62	5.62	5.62	
costs		5.990	6.900	6.841	
number of steps		14	18	18	
distance calculations		403710	13596	41230	
search time [s]		16.607	0.169	4.347	



(a) Results of *2D Pre-Planning*. (b) Step sequence for shortest path.

Fig. 8: Results of pre-planning and final step sequence for complex environment. Large obstacles in dark grey, safety zones (different for step planner and mobile platform planner) in light grey, shortest path in green, and other paths in blue.

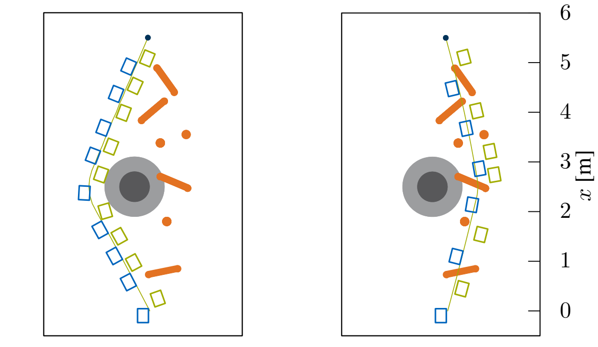
The next test environment is depicted in Fig. 8b. Even in this complex environment our *2D Pre-Planning* was able to find multiple path variants (see Fig. 8a). The final step sequence for the shortest path is depicted in Fig. 8b. To investigate the benefit of finding multiple path variants, another test case with small obstacles is presented. The *2D Pre-Planning* finds two possible path variants and corresponding step sequences as shown in Fig. 9. The results are summarized in Tab. II. When comparing the path costs the right solution yields higher costs even though its 2D path is shorter than the left one. This increase is caused by the presence of small obstacles, which result in more expensive steps.

TABLE II: Results of a test case with two alternative solutions for step sequences (see Fig. 9).

	Solution 1 (left)	Solution 2 (right)
length of 2D–path [m]	5,647	6,036
costs	6,213	6,337
number of steps	17	11
distance calculations	1557	170810
search time step–planner[s]	0,0415	24,99

B. Validation in Experiments

We validated our methods in multiple experiments with the bipedal robot *Lola*. The experiments includes non-traversable



(a) Longer 2D path (Solution 1) (b) Shorter 2D path (Solution 2)

Fig. 9: Step sequences for two different 2D–paths.

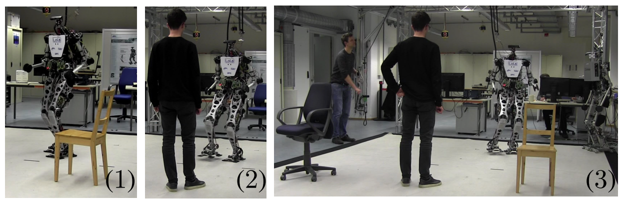


Fig. 10: Experiments: (1) Static environment with large obstacle, (2) Dynamic environment with moving human, (3) Dynamic environment with moving human and multiple large obstacles.

traversable obstacles (see Fig. 3). Furthermore, we successfully conducted experiments in dynamic environments with a moving human and multiple static obstacles. A sequence of snapshots of our experiments is depicted in Fig. 3 and Fig. 10. [4] The experiments validated the real-time capability of our methods: the environment in the experiments is completely unknown and dynamic. It is modeled using our on-board vision system. Fig. 11 shows the result of the mobile platform planner and the step planner for three different steps of the experiment with a dynamic environment with a moving human and multiple large obstacles. Since the field of view is limited, *Lola* perceives a new part of the environment with each step. Furthermore, *Lola* has to react to a human dynamically stepping in its way. This is clearly visible in the differences of the environment of snapshots $1-3$ in Fig. 11. The mobile platform planner adapts to the changing environment and determines executable 2D paths. The step planner uses these 2D paths as a guideline and calculates step sequences. Fig. 11 emphasizes the real-time character of the step planner. It is aborted after $n_{exp,des}$ steps are calculated or the time limit is reached.

VI. Discussion, Summary and Outlook

A. Discussion

The presented methods influence the A*–search and may jeopardize the optimality character of the A*–search. If we assume that the *2D Pre-Planning* finds all possible 2D paths,

[4]A video of our experiments is available online https://youtu.be/-VvxzFg9ATU.

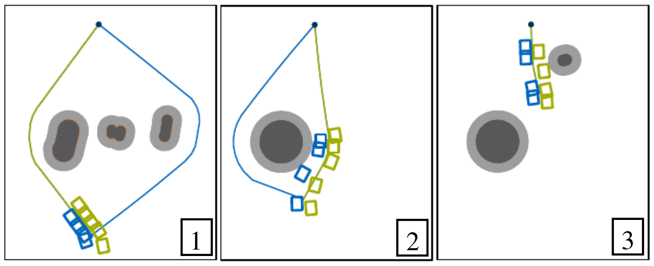


Fig. 11: Experiment in dynamic environment with moving human and multiple large obstacles (refer to Fig. 10-(3)): calculated step sequences and environment approximation at three different instances.

including the shortest path, in the reduced map: (1) using the shortest 2D path as initial estimate will always underestimate the distance of the path subsequently optimized by the step planner and will therefore not influence the optimality of the result of the A*–search. Furthermore, the shortest path approximates the remaining path costs better than a Euclidean distance. Therefore, it accelerates the search, but will always underestimate the remaining costs. (2) Limiting the search area of the A*–search and reducing the analyzed step suggestions influences the optimality of the A*–search. It will still find an optimal solution, but only in the reduced search area. Since, the search space is limited, it is important that the *2D Pre-Planning* determines more than one solution. In the presence of multiple obstacles not considered in the reduced map, a 2D path may not be walkable. Searches following different 2D paths significantly increase the probability to find an executable step sequence. (3) Calculating φ based on the 2D paths reduces the dimension of the search space by one. Therefore, the quality of the calculated step sequence will always be inferior to the solution of the search in a higher dimensional search space.
In conclusion, only the integration of the 2D path as a heuristic in the A*–search does not decrease the solution's optimality. For real-time application, we gain the advantage of significantly faster calculation times in the case of sub-optimal solutions when limiting the search space or reducing the dimension of the search space. With the application of legged robots in interaction with human users in mind, the method's evaluation has another facet. A robot trying to follow a 2D path instead of executing an internally calculated and optimal step sequence may help a user to predict the robot's behavior and interact with it. Applying the methods presented in Subsection IV-B, the search for an optimal step sequence could not only be improved, but a user has another option to guide a robot, for example by a 2D path as input.

B. Summary and Outlook

In this paper, we propose strategies to use a novel mobile platform planner for improving navigation of bipedal robots. We reduce the full map of the environment to a map including only obstacles the robot is not able to traverse. We introduce a mobile platform planner which is able to find a set of solutions instead of only one path. This is important for our application for two reasons: (1) A 2D path may not be traversable or sub-optimal because of the decreased information value of the reduced map. (2) For real-time application, it is advantageous to search for multiple paths at the same time on different cores. With a limited processing power, having multiple possibilities to choose from makes navigation in real-environments more robust. Moreover, we present three different methods to combine 2D paths with an A*-based step planner. They may use the result of any mobile platform planner. On the one hand, these methods accelerate significantly the search for step sequences in the presence of non-traversable areas. On the other hand, it is possible for a user to interact with the robot. The user could provide a 2D path the robot would follow. We analyzed our methods in simulation and validated their real-time character in experiments with our robot *Lola*. In future projects, we plan to parallelize multiple A*-searches and to evaluate their strength in more difficult environments.

References

[1] T. Buschmann, "Simulation and Control of Biped Walking Robots," PhD thesis, Technical University of Munich, 2010.

[2] K. Nishiwaki, J. Chestnutt, and S. Kagami, "Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1251–1262, Aug. 2012.

[3] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot - 3rd report: Dynamics error compensation-," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2009, pp. 1594–1600.

[4] A.-C. Hildebrandt, D. Wahrmann, R. Wittmann, D. Rixen, and T. Buschmann, "Real-Time Pattern Generation Among Obstacles for Biped Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[5] J. Chestnutt, "Navigation Planning for Legged Robots," PhD thesis, Carnegie Mellon University, 2007.

[6] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *IEEE-RAS International Conference on Humanoid Robots*, 2012.

[7] P. Karkowski and M. Bennewitz, "Real-Time Footstep Planning Using a Geometric Approach," in *IEEE International Conference on Robotics and Automation*, 2016.

[8] A. Hornung and M. Bennewitz, "Adaptive Level-of-Detail Planning for Efficient Humanoid Navigation," in *IEEE International Conference on Robotics and Automation*, 2012.

[9] Y. Ayaz, K. Munawar, M. B. Malik, A. Konno, and M. Uchiyama, "Human-like approach to footstep planning among obstacles for humanoid robots," *International Journal of Humanoid Robotics*, vol. 4, no. 01, pp. 125–149, 2007.

[10] T. Buschmann, S. Lohmeier, M. Schwienbacher, V. Favot, H. Ulbrich, F. von Hundelshausen, G. Rohe, and H.-J. Wuensche, "Walking in Unknown Environments - A Step Towards More Autonomy," in *IEEE-RAS International Conference on Humanoid Robots*, 2010.

[11] D. Wahrmann, A.-C. Hildebrandt, R. Wittmann, D. Rixen, and T. Buschmann, "Fast Object Approximation for Real-Time 3D Obstacle Avoidance with Biped Robots," in *IEEE International Conference on Advanced Intelligent Mechatronics.*, 2016.

[12] A.-C. Hildebrandt, M. Demmeler, R. Wittmann, D. Wahrmann, F. Sygulla, D. Rixen, and T. Buschmann, "Real-Time Predictive Kinematic Evaluation and Optimization for Biped Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[13] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous robots*, vol. 13, no. 3, pp. 207–222, 2002.

[14] T. Ersson and X. Hu, "Path planning and navigation of mobile robots in unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[15] P. K. Kim, Jin-OhKhosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE International Conference on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, 1992.

[16] M. Schwienbacher, T. Buschmann, S. Lohmeier, V. Favot, and H. Ulbrich, "Self-Collision Avoidance and Angular Momentum Compensation for a Biped Humanoid Robot," in *IEEE International Conference on Robotics and Automation*, 2011.

[17] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *IEEE International Conference on Robotics and Automation*, 1993.