This is a preprint of a paper that appeared in the Proceedings of
IEEE-RAS International Conference on Humanoid Robots, pp. 287-294, Nov 18-20, 2014

Supervised Footstep Planning for Humanoid Robots in Rough Terrain Tasks using a Black Box Walking Controller

Alexander Stumpf*, Stefan Kohlbrecher*, David C. Conner† and Oskar von Stryk*

***Abstract*— In recent years, the numbers of life-size humanoids as well as their mobility capabilities have steadily grown. Stable walking motion and control for humanoid robots are already well investigated research topics. This raises the question how navigation problems in complex and unstructured environments can be solved utilizing a given black box walking controller with proper perception and modeling of the environment provided. In this paper we present a complete system for supervised footstep planning including perception, world modeling, 3D planner and operator interface to enable a humanoid robot to perform sequences of steps to traverse uneven terrain. A proper height map and surface normal estimation are directly obtained from point cloud data. A search-based planning approach (ARA*) is extended to sequences of footsteps in full 3D space (6 DoF). The planner utilizes a black box walking controller without knowledge of its implementation details. Results are presented for an Atlas humanoid robot during participation of Team ViGIR in the 2013 DARPA Robotics Challenge Trials.**

I. Introduction

There has been significant progress in humanoid robotics research in recent years. Advances both in hard- and software raise the question how humanoids can be used to assist or replace humans in hazardous environments. Motivated in part by the lack of suitable robotic response technologies at the Fukushima Daiichi nuclear disaster, the DARPA Robotics Challenge (DRC)[1] for disaster response scenarios aims at answering this question. A crucial capability for the use of humanoids in disaster environments is the ability to traverse different types of terrain in harsh environments. Examples of such terrain are uneven ground, narrow doorways and ladders, all of which are also part of the capabilities that are tested within the DRC. The first evaluation of the participating teams was done at the DRC Trials in December 2013 with real robots where the presented approach was already used.

The unpredictability of disaster scenarios implies that they are among the robotic applications that benefit the most from the cognitive abilities of a human operator. For this reason, an approach that leverages the complementary abilities of a human operator and a robotic system is likely to perform best in such situations.

Towards this end, we present a footstep planning system that allows supervised locomotion and navigation in uneven environments including perception, world modeling, 3D planning and operator interaction. The system uses the perception system of the robot to generate a 3D world model and 3D step sequences. Operators can thus select target poses for the robot system without having to care about terrain geometry or even single footstep placement. The implementation of the presented planning system is part of the Team ViGIR's approach [1] to the DRC Trials 2013.

*Department of Computer Science, TU Darmstadt
stumpf,kohlbrecher,stryk@sim.tu-darmstadt.de
†TORC Robotics conner@torcrobotics.com
[1]http://theroboticschallenge.org

II. Related Work

Humanoid locomotion is still a challenging task even though walking capabilities of humanoid robots have increased significantly in the recent years. While walking over flat surfaces is relatively easy, the requirements for unstructured terrain are much higher. The walking controller has to face many external disturbance and the robot is more likely to slip on non-flat surfaces. In [2][3] this issue is addressed by implementing dynamic pattern generators. However, both implementations do not cover navigation. This problem might be mitigated by using a 3D planner that provides suitable foot positions for the walking controller to follow. In this way the walking controller can be informed about the underlying terrain in advance.

The *Covariant Hamiltonian Optimization for Motion Planning* (CHOMP) [4] is a trajectory planner which optimizes a given initial trajectory with respect to a cost function and avoiding collisions. Another approach by Schulman et al. [5] uses sequential convex optimization for optimal trajectory generation. For this purpose each criterion, such as ZMP-stability, can be modeled as as a constraint for optimization. All approaches are capable to solve rough terrain tasks with respect to robot kinematics, but they need an accurate robot model including kinematics and masses and solving the optimization problem may be computationally expensive. Furthermore they can not use existing walking controllers well because they overfit the solution towards a given robot model, which is likely not the same as the used model by the existing black box walking controllers.

Another category are contact-before-motion approaches which decouple the planning and motion layer. Bouyarmane et. al. introduce a multi-contact-planning approach in [6][7] which is designed to be used with whole-body-control (WBC). This approach generates a sequence of contact configurations which have to be executed by the WBC. It requires significant computational resources and so far is not feasible for use in tasks such as encountered in the DRC, as they do not allow for extended computation time.

As human resources and development time during the DRC competition are very limited, we require an approach which can be integrated in our systems easily and uses

existing software. Thus, it should have a Robot Operation System (ROS)[2] interface and be able utilize existing walking controllers. Finally we decided to use a graph-based footstep planner similar to one that was already used successfully for Asimo in the past [8]. This approach generates a sequence of footsteps which can be used by walking controllers. Hornung et. al. [9][10] implemented an open-source version which is available for ROS, enabling quick integration in our system.

In [11][12] a first extension of previous implementation of Hornung et. al. footstep planner [9][10] is shown. In [12] they introduce how to generate a height map online using an onboard stereo camera system and a collision checking method using pre-generated inverse height maps of each action. Although they generate a 3D world model, their approach is only extended by special actions for step up and step down motions. This enables 3D planning on flat ground using predefined actions but is not sufficient for real world application due to missing capability for planning on sloped or random terrain.

Another open question is how to use the robot walking controller capabilities in rough terrain tasks. This question was addressed by the *Learning Locomotion Project* hosted by DARPA. All participating teams implemented similar approaches using a graph-based planner with learned or optimized cost functions [13][14][15][16] which shows that learning cost functions is a valid option.

In contrast to search-based planning approaches, MIT recently presented their footstep planning system [17] which generates a footstep plan by continuous optimization. For this purpose they implemented a mixed-integer quadratically-constrained quadratic program (MIQCQP). They demonstrate that their planning system is able to solve complex terrain scenarios but world modeling itself is still a weakness in their approach. In the presented work the operator has to define obstacle free regions manually by placing polygons.

Summarizing, only a few authors address terrain modeling in their work e.g. generation of height maps from stereo camera data [18][12]. Most approaches are based on external motion tracking systems or assume a known world model. This aspect may not be neglected if the robot should walk in real-world environments, which motivates our work.

III. Experimental Platform

The proposed approach is designed for humanoid robots providing a walking controller. Thus, the presented work is evaluated with the Atlas robot without limitation to generality. Atlas is a 1.88m tall with a weight of 150kg near anthropomorphic robot developed by Boston Dynamics Inc. (BDI) as the direct successor to PETMAN [19]. The main external sensor is a Carnegie Robotics Multisense SL sensor mounted as the head. This sensor uses both a Hokuyo UTM-30LX-EW LIDAR mounted on a slip ring for continuous rotation and a stereo camera system. Furthermore the robot provides a pose estimate based on a internal IMU and internal joint sensing.

[2]http://www.ros.org/

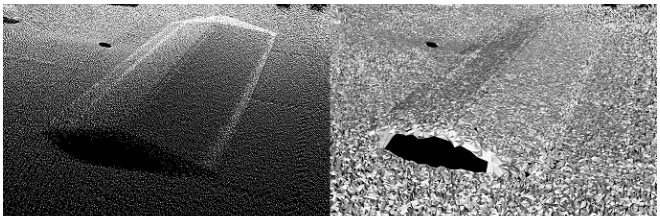
The robot is delivered with a proprietary walking controller developed by BDI. The closed source "BDI Walking/Stepping Behaviors" provide several controller modes including a quasi-statically stable stepping mode as well as dynamically stable walk mode. From our perspective, the BDI Walking Behavior is a pure black box; this emphasizes the generality of our footstep planning approach.

IV. World Modeling

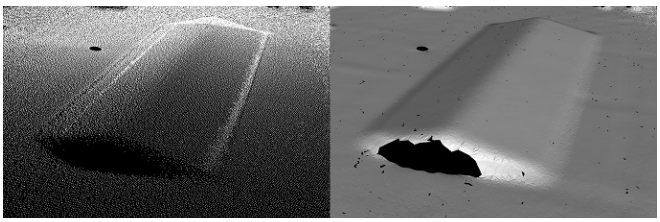
To our knowledge graph-based planners have nearly only be used for planning in flat environments. In contrast 3D planning needs the ability to estimate the full 3D state of each placed foot. This concerns in particular the roll, pitch and height of the foot pose which are not needed for 2D planning approaches.

Thus, a suitable world model is essential for reliable footstep planning in rough terrain scenarios and has to provide all required information in an efficient way. Although Atlas is equipped with multiple sensor systems, only the LIDAR scanner is a suitable for large range scans providing accurate terrain data for long distance planning. For this purpose a processing pipeline was implemented to generate a suitable word model for rough terrain planning. The first step is to aggregate each single LIDAR scan into a 3D point cloud given an aggregation horizon in seconds. In our case we use only data which was collected since the robot has not moved due to missing robot state estimation and correction.

Afterwards, noise is filtered out from the resulting point cloud by applying the *Voxel Grid* and *Moving Least Square* filters of the *Point Cloud Library* (PCL)[3]. Additionally, a *Pass Through* filter is used to reduce the computational expense by cropping the region of interest. All parameters of the filtering process were adjusted offline by comparing the resulting surface reconstruction of the *Greedy Projection* approach which is provided by PCL. Figure 1 emphasizes the difference between noisy and filtered point clouds.



(a) Noisy point cloud and 3D surface reconstruction due to missing filtering



(b) Filtered point cloud and 3D surface reconstruction

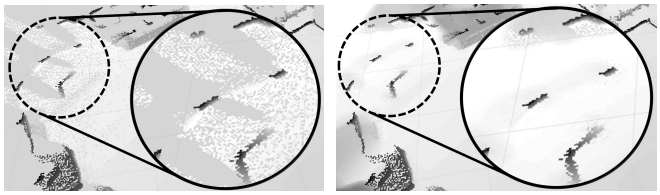
Fig. 1: Examples for 3D surface reconstruction

[3]http://www.pointclouds.org/

Given the smoothed point cloud we can extract the height and normal of the surface. The height map is obtained easily by storing the z-component of each point of the cloud in a discretized 2D grid map. Each cell of the height map contains the highest value discovered in the point cloud. Unfortunately the height map contains many small holes due to sparse LIDAR data. This issue may let the planner fail unnecessarily due to missing height information. Therefore, missing height information $\hat{p}_z$ is estimated by the weighted average of the point's $\hat{p} = (\hat{p}_x, \hat{p}_y, \hat{p}_z)^T$ k-nearest neighbors $p^{(i)} = (p_x^{(i)}, p_y^{(i)}, p_z^{(i)})^T$ in the point cloud:

$$\hat{p}_z = \frac{1}{k}\sum_{i=1}^{k} p_z^{(i)} \cdot \left(1 - \frac{\left\| \begin{pmatrix} p_x^{(i)} \\ p_y^{(i)} \end{pmatrix} - \begin{pmatrix} \hat{p}_x \\ \hat{p}_y \end{pmatrix} \right\|^2}{\sum_{j=1}^{k} \left\| \begin{pmatrix} p_x^{(j)} \\ p_y^{(j)} \end{pmatrix} - \begin{pmatrix} \hat{p}_x \\ \hat{p}_y \end{pmatrix} \right\|^2} \right) \quad (1)$$

which is demonstrated in figure 2.



(a) Cluttered pixels indicate missing height information

(b) Homogeneous height data is now available due to estimation

Fig. 2: Example for gap filling in a height map

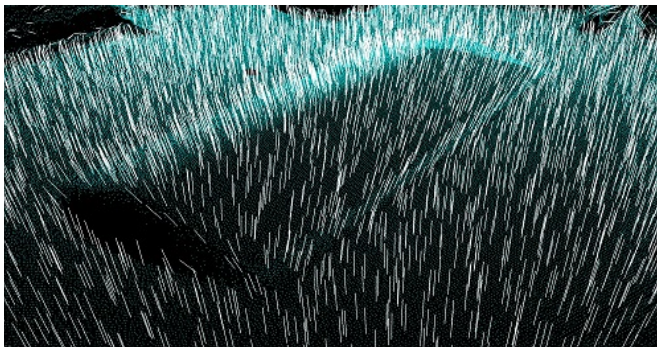


Fig. 3: Resulting normal estimation of a pitch ramp using PCA-based approach: The white lines visualize the estimated normals while the underlying point cloud consists of the turquois points.

The normal is extracted by applying a Principle Component Analysis (PCA)[4] based approach from PCL on the neighborhood of the query point. This operation is performed on every point of the cloud which is surrounded by a sufficient amount of neighbors in a given distance. As the solution for the direction is not unique, all normals are converted to point upright relative to gravity (see figure 3). Given a target position and orientation the normal can be obtained and used to determine the roll and pitch angle for this query point. In order to determine a normal for any query point efficiently, all of them are saved in a k-d-tree, which is already implemented in PCL. In our case we use the position as query key and the normal is resolved with $O(\log n)$ in average.

For simple body collision checks the LIDAR scans are additionally used to generate an octomap [20] which results in a 3D world representation composed of cubes. This octomap is used to generate 2D occupancy grid maps by projecting a slice of the octomap.

V. Footstep Planning

Our novel approach efficiently generates suitable plans for rough terrain tasks using existing black box walking controllers without deep knowledge about their implementation that requires decoupling of planning and execution layers. For this purpose the planner must be able to determine feasible paths with the given world model as fast as possible to operate the robot efficiently.

The presented work is based on Hornung's footstep planning framework [9][10][11]. In this paper we want to highlight the major extensions and all necessary steps to perform footstep planning for rough terrain tasks under real-world applications. Finally, the approach is evaluated with the Atlas robot but may be easily transferred to general classes of humanoid robots providing a walking controller.

The basic footstep planner by Hornung et. al. defines the state $\boldsymbol{s}$ as the pose of the foot $\boldsymbol{s} = (x, y, \theta, f)$, where $\{x, y\}$ denote the 2D-position, θ the orientation and $f \in \{left, right\}$ the corresponding foot. All successor states $\boldsymbol{s'}$ of $\boldsymbol{s}$ are generated by applying the transition function $t(\boldsymbol{s}, a)$ where $a \in A$ are actions describing the displacement vector $(\Delta x, \Delta y, \Delta\theta, f)$ and f denotes the supporting foot. Therefore the successor state $\boldsymbol{s'}$ is simply derived by:

$$\boldsymbol{s'} = t(\boldsymbol{s}, a) = \boldsymbol{s} \overset{a}{\mapsto} \boldsymbol{s'} \quad (2)$$

The set A consists of all possible actions a which are also denoted as "footstep primitives". For simplicity we omit the foot f in all following representations as we assume that the planner will generate an alternative sequence of footsteps and every action can be mirrored in following way: $(\Delta x, \Delta y, \Delta\theta, left) = (\Delta x, -\Delta y, -\Delta\theta, right)$.

Finally this representation is used to generate a graph with states as nodes and actions as transition condition between the nodes. Here, the Anytime A* (ARA*) [21] algorithm is used to determine the shortest path from start to goal which is transformed to a sequence of footsteps afterwards.

A. States, Actions and Transition Model

The key parts of graph-based planning are the states, actions and the transition model. These components have to be adapted in a way they become suitable for rough terrain tasks requiring full 3D planning space.

[4] http://pointclouds.org/documentation/tutorials/normal_estimation.php

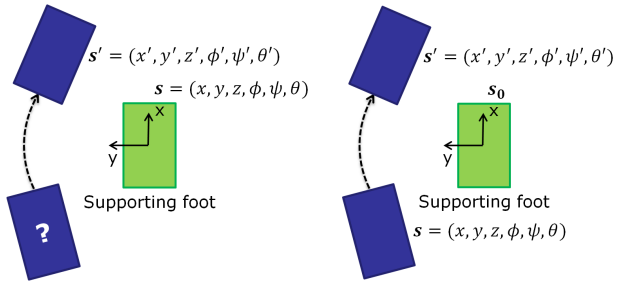
1) States: Rough terrain traveling requires a 3D state space representation. Thus, the original approach was extended by z, ϕ (roll) and ψ (pitch) to the 3D state space $\boldsymbol{s} = (x, y, z, \phi, \psi, \theta) = (s_x, s_y, s_z, s_\phi, s_\psi, s_\theta)$. Fortunately, these new components are constrained by the underlying surface and can easily be obtained by the generated height map and normals as described in section IV. For this purpose the corresponding normal $\boldsymbol{n} = (n_x, n_y, n_z)^T$ has to be converted to Euler angles given the target orientation θ of the foot:

$$\phi = -\arcsin(n_x \sin(-\theta) + n_y \cos(-\theta)) \quad (3)$$

$$\psi = \arcsin(n_x \cos(-\theta) - n_y \sin(-\theta)) \quad (4)$$

which fits a plane tangential to the surface at the query point. The Euler angle representation simplifies the comprehension and design of cost functions. Although the extended state space now has six DoF, the search space itself remains at three DoF. For this reason the computational effort for searching an optimal solution is not increased by this modification, provided the world model can be queried efficiently.

With the growth of larger, more complex humanoid robots, consideration of dynamics has become more important and cannot be neglected for the Atlas robot. For this reason the cost functions were extended to evaluate the cost $c(\boldsymbol{s}, \boldsymbol{s'})$ for full step representation (see Figure 4b) instead of the original half step representation (see Figure 4a) where the origin of the swing foot is unknown. This allows us to consider the dynamic behavior estimate of the robot executing this step sequence. We define a new representation for step sequences consisting solely of the start state $\boldsymbol{s}$ and goal state $\boldsymbol{s'}$ of the moving foot which are given relative to supporting foot $\boldsymbol{s_0}$. This kind of representation enables a unique representation for any possible step, and permits the use of lookup tables instead of computational expensive function evaluations, to reduce planning time significantly.



(a) half step representation (b) full step representation

Fig. 4: Modification of step representation

2) Actions: Although the original approach was designed for small humanoid robots walking on flat ground, the search space remains the same (see Section V-A.1). Thus, an action a is still defined as displacement vector $(\Delta x, \Delta y, \Delta\theta)$. Rough terrain tasks demand higher accuracy in planning level due to unstructured debris on the floor. This problem is further exacerbated by larger humanoid robots whose reachable region $\mathcal{R}$ for a single step is significantly increased. Therefore, the action set A must be increased as well to maintain a sufficient variation of discrete actions and accuracy for crossing rough terrain. For this reason it is not feasible to extend the set A manually like the authors in [12] do. Thus, we define a reachability polygon $\mathcal{R}$ next to the supporting foot $\boldsymbol{s_0}$ from which the footstep primitive set A is sampled in a discrete way.

B. Cost Functions and Heuristics

The cost function used in combination with a heuristic has the maximum impact on the planner solution [10]. This makes designing cost functions and heuristics the most important and challenging task.

1) Cost functions: As we do not have any detailed insight into the robot's walking controller (here the *BDI Walk Behavior*) all cost functions can only be designed by systematic experiments, expertise or machine learning approaches. Besides the robot capabilities basic properties like minimal number of steps or shortest path has to be considered by cost functions too. Additionally, life-size humanoid robots have versatile walking capabilities which should be utilized in rough terrain, so the footstep planner has to minimize multiple, possibly competing costs e.g. shortest path and minimal fall risk. For this reason a hierarchical system of cost functions was implemented, dividing each criterion in a different layer which is finally conquered individually by a specific cost function. This design was realized by using the Decorator Pattern [22], enabling the composition of cost functions in a flexible manner.

In this section we will only introduce briefly the implemented cost functions.

- *Constant*: This functions adds the constant value c_{step} on top of all cost regardless of the given step. Using this function enforces the planner to generate a step-minimal solution.
- *Euclidean*: The euclidean cost function computes the 2D distance traversed by the robot's torso. This way the planner is enforced to find the shortest path.
- *GPR*: This cost function wraps a *Gaussian Progress Regression* implementation estimating step cost by previously collected and offline learned observation.
- *Map*: In Section V-A.1 we have pointed out that instead of evaluation of computational expensive cost function a simple lookup table may be used which is implemented by this cost function,
- *Boundary*: This cost function implements the robot specific capabilities and invalidates all step sequences which can probably not performed by the robot. This cost function was determined by systematic experiments with the real robot to discover the limits of the system which are tightened to decrease the risk of failure. In sum, this cost function penalize violation of these tightened limits and has the most influence on $\hat{A}(\boldsymbol{s})$.
- *Dynamics*: The dynamic behavior e.g. accelerations are modeled as additional cost and prevents high accelerations of the robot torso.

- *Ground Contact*: This cost function utilizes the ground contact estimation (see Section V-C) and adds artificial cost, when the given step position does not fit the underlying terrain perfectly.

In the current state following hierarchy is used (top-down): *Constant*, *Euclidean*, *Boundary*, *Dynamics* and *Ground Contact*. Finally the accumulated cost represents the effort to execute the step sequence.

2) Risk Measurement: The reachability polygon $\mathcal{P}$ introduced in Section V-A.2 implies the risk of including bad footstep placements. For this reason we introduce in our novel approach a quantity denoted as risk to distinguish between effort (cost) and feasibility (risk) of a step. Merging risk and cost in a single value would not prevent the planner from using bad steps in a plan when only a small number of step options are available. Furthermore, risk evaluation allows to generate dynamically a subset of valid actions $\tilde{A}(\boldsymbol{s}) \subseteq A$ depending on current state $\boldsymbol{s}$ which improves the quality of generated plans. Analogous to evaluation of cost, the risk is simultaneously computed in every layer. Thus, feasibility of a step can now be determined based on the accumulated risk while the effort to execute this step is represented solely by the cost.

3) Heuristic: Finding the global best solution is not guaranteed by using the ARA*-planner in *anytime* mode. While the cost function defines the shape of the resulting (local) cost-minimal plan, the heuristic influences which local minimum is found by the planner; thus, the heuristic may not be neglected. The original heuristics uses

$$h(\boldsymbol{s}) = \|\boldsymbol{s} - \boldsymbol{s_{goal}}\| + c_\theta \cdot |\Delta\theta| + c_{step} \cdot \frac{\|\boldsymbol{s} - \boldsymbol{s_{goal}}\|}{d_{max}} \quad (5)$$

to estimate remaining cost with d_{max} as maximal step distance, $\Delta\theta$ the shortest angle between θ, θ_{goal} and c_θ, c_{step} as predefined cost per angular difference or step. In combination with the cost functions defined in Section V-B.1 the planner has a poor planning performance (see figure 5a), because the heuristic does not take into account that the maximum step distance d_{max} should depend on the direction of movement. Hence, we distinguish between transversal Δx_{max} and lateral Δy_{max} step distance limits leading to following modified heuristic:

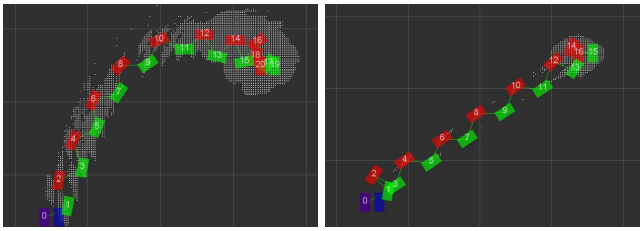
$$n_{steps} = \lfloor \frac{\Delta\hat{x}}{\Delta x_{max}} + \frac{\Delta\hat{y}}{\Delta y_{max}} \rfloor \quad (6)$$

$$\hat{h}(\boldsymbol{s}) = \|\boldsymbol{s} - \boldsymbol{s_{goal}}\| + c_\theta \cdot |\Delta\theta| + c_{step} \cdot n_{steps}, \quad (7)$$

where $\Delta\hat{x}$ and $\Delta\hat{y}$ denote the needed transversal and lateral movement distance without using any rotational movement. Thus, n_{steps} is just an estimation of minimal number of total transversal and lateral steps needed to reach the goal. In usual cases $\Delta x_{max} > \Delta y_{max}$ holds which means the robot can move faster forwards than sidewards. Thus, the modified heuristic $\hat{h}(x)$ prefers moving directly towards the goal why $\hat{h}(x)$ performs better than $h(x)$ in Figure 5b.

C. Collision Check

Rough terrain tasks demand the ability to detect obstacles and walkable surfaces and utilize these information for navigation. Our approach is focused on solving the terrain task of the DRC Trials which has fairly sloped terrain without walls and the robot doesn't need to perform multi-contact-planning. Thus, we may assume if two sequent states are collision free then the trajectory will be collision free too.



(a) Sight line euclidean step cost heuristic: 101.659 expanded states, 95s planning time

(b) Modified euclidean step cost heuristic: 10.102 expanded states, 9s planning time

Fig. 5: Comparison of heuristics

1) Occupancy Grid Map: Hornung et. al. already implemented an efficient collision check strategy for rectangular bodies like the feet which is proposed by Sprunk et al. [23] using an occupancy grid map. They assume that all not surmountable obstacles are artificially inflated in the occupancy grid map [10] to keep the upper body collision free. But this approach is insufficient for our purposes because the robot has to traverse narrow doors. Therefore, we split collision checking into two layers: One layer is exclusively responsible for upper body and the other for feet collision check. Analogously, we extend the footstep planner to use different occupancy grid maps for each layer which are generated by slicing the octomap as described in Section IV.

2) Ground Contact Estimation: In the domain of 3D planning collision checking using 2D occupancy grid maps has the drawback to take not the terrain height into account. When using the occupancy grid map, we can not distinguish between holes and hills. Furthermore, while performing as expected, occupancy grid maps have the drawback of wasting space by encircling each obstacle with non-traversable cells. As a result, collision checks with occupancy grid maps are too strict because they enforce placing each foot avoiding obstacles completely. This results in longer paths and the ARA*-planner takes a lot of more computational time trying to find shorter solutions. Furthermore the occupancy grid map based collision checks do not allow any kind of overhang when climbing upstairs.

This motivates our new approach to use solely the height map introduced in Section IV. Given a target foot state $\boldsymbol{s}$ the undersurface of the footprint is sampled equally spaced to compute the vertical distance $\Delta z = h(s_x, s_y) - s_z$ relative to the terrain surface $h(x, y)$ which is shown in Figure 6. Each sampling point is classified by using thresholds into *collision* ($\Delta z > 0.75cm$), *overhung* ($\Delta z < -1.0cm$) or otherwise *contact*. If at least one sampling point is classified as *collision* the foot state will be treated as invalid. Otherwise the ground contact support is given as percentage of sampling points classified as *contact*. These narrow thresholds are feasible

because of well filtered LIDAR data (see Section IV) and all surfaces in the competition are flat. In general the thresholds must be scaled towards data quality. The ground contact support estimate becomes worse with increasing noise in case of many overhanging points and the planner performance decreases due to many false positive collision detections. In future work we plan to use the convex hull of *contact* points as ground contact estimation that promises more robustness against false positive *overhung* classifications. Furthermore, it enables the planner to generate steps having only at least three *contact* points (e.g. stepping on three poles). An estimate of surface friction is neglected for now because all tasks are built up with slip-proofed surfaces. But friction estimation will be added when required by future competition.

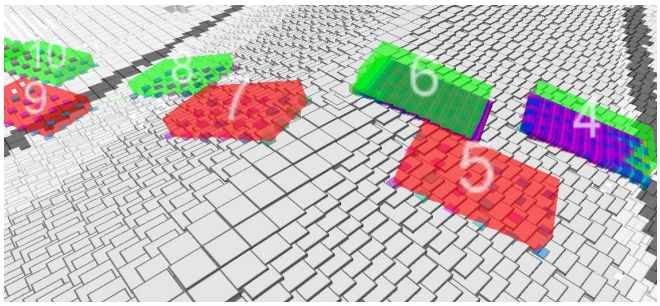
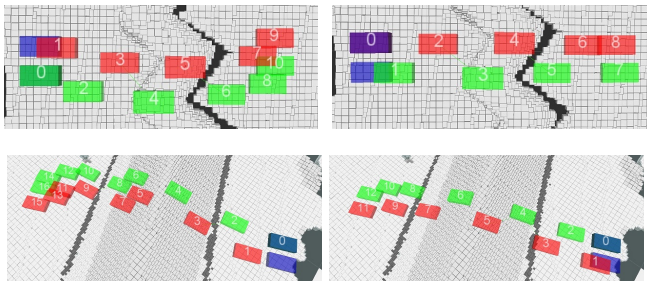


Fig. 6: This example shows the sampling points at the undersurface of the footprint. The steps 4 and 6 have been sampled in higher resolution.

The ground contact estimation superfluous the usage of discretized occupancy grid maps. Furthermore the planner is now able to plan overhanging steps which increases flexibility of foot placement. For this reason the planner is able to find and optimize solutions quicker improving significantly the resulting plan. In Figure 7 two examples of the DRC Trials are given where the ground contact support outperforms the usage of occupancy grid maps by generating shorter and straighter paths.



(a) occupancy grid map (b) ground contact estimation

Fig. 7: Comparison of collision checking approaches: In both examples the ground contact estimation outperforms the occupancy grid map.

D. Miscellaneous modifications

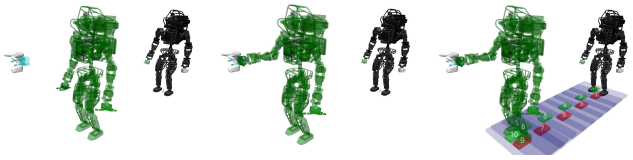
1) Start foot selection: As the first step is not explicitly defined by the planning request, the planner has to select the starting foot by itself. The original footstep planner by Hornung et. al. selects always the left foot. In contrast we add the logic to select the foot which is closer to the goal pose. This behavior allows the planner to turn quicker towards the goal position which results in shorter paths.

2) Automatic goal pose refinement: The operator may define the planning goal by pose. In this case the planner itself has to define the final feet position at the end of the plan. The planner is able to autonomously project both feet on the sloped surface. If the initial feet configuration collides, the planner is able to shift slightly the feet pose to a collision free configuration within the given reachability polygon $\mathcal{R}$.

3) Post-Processing: Walking controllers are supposed to define further parameters for each step e.g. lift height. We are interested in getting as fast as possible a solution, so optimizing these parameters during planning process is not feasible due to increased dimension of search space. Thus, these parameters are improved in a post-process step after the final solution was found. In the current state we determine only the lift height by searching the maximum terrain height in sight line between origin and target pose of the step.

VI. Operator Interaction

For supervised robot operation different abstraction layers for controlling the footstep planner is available. With increasing task complexity a more detailed operator interface can be used. The most simple way to interact with the footstep planner is by moving the ghost robot to the desired goal position demonstrated in Figure 8. Alternatively, the operator may define directly a target pose by clicking on the grid map.



(a) (b) (c)

Fig. 8: Use case of footstep planning: The operator places a template (a). The ghost robot (green) will be aligned with the template (b) and the footstep plan is generated (c).[24]

For more complex task like rough terrain several widgets are available which we want to introduce briefly:

- *Parameter Widget*: This widget provides access to the major planner parameters.
- *Step Widget*: The widget provides the option to define manually single steps or whole patterns.
- *Terrain Request Widget*: 3D terrain model generation is unfortunately computational expensive; thus, it should only be generated by operator request.

Transmission bandwidth is very limited during the competition. Thus, the operator receives only a thinned out version of the point cloud consisting of a very small fraction of the original point cloud data to verify the generated footstep plan which is visualized with boxes which can be seen in Figures 6 and 7.

VII. Results

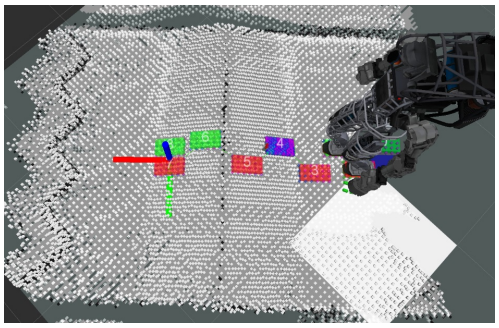


Fig. 9: OCS perspective of the ramp (DRC Trials)

Fig. 10: Robot traversing the pitch ramp (DRC Trials)

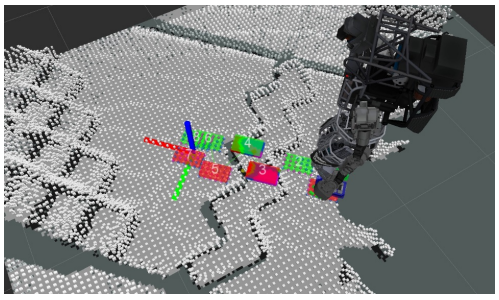


Fig. 11: OCS perspective of the chevron hurdle (DRC Trials)

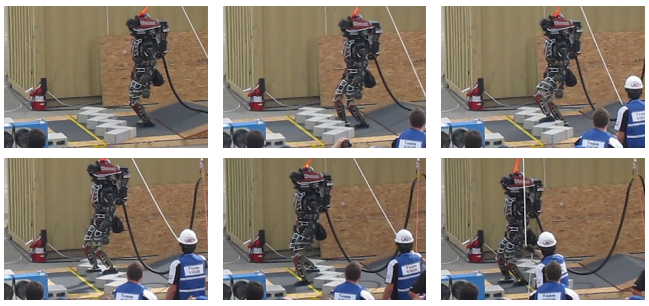


Fig. 12: Robot traversing the chevron hurdle (DRC Trials)

At an early development stage we tried to learn a suitable step cost function using Gaussian Progress Regression (GPR) for the simulated robot. For this purpose random step pattern has been generated and executed in simulation. After each experiment the step pattern was rated by successful execution or fall. In total over 40k experiments were performed in simulation whose data was used to learn a step cost function which estimates the risk of failure. In `https://www.youtube.com/watch?v=hT-_n4icg54` the footstep planner is able to utilize the black box walking controller much better compared to the manual defined footstep primitives in `https://www.youtube.com/watch?v=JMCREHzimpM`. However, it turned out that offline learning approaches like GPR takes too much effort to train. All recorded data and the learned cost functions are supposed to be invalidated every time the robot is modified, which happened often because Atlas was still in development. Thus, extensive (re)training must be performed, which is not feasible for a real robot system. Furthermore, simulation-based learning is not feasible because of significant differences between the model and real robot.

As we got only a limited developing time until the DRC Trials, we had to investigate in an approach which is easier to maintain in case of changes of the robot system or the walking controller. For this reason the final solution is to define a hierarchical step cost function determined and optimized by systematic experiments (see Section V-B.1).

This approach was successfully used during the DRC Trials in all tasks without any falls caused by the footstep planner. While the BDI walking controller provided a very stable walk, we demonstrated how to utilize this walking controller close to the limits. During the DRC Trials almost all teams using the Atlas robot with the provided BDI walking controller performed significant smaller step sizes and with it slower travel speed.

The terrain task was the supreme task for locomotion where the planner performed well. It took only a few minutes and very few interaction steps by the operator to cross the pitch ramp and the chevron hurdle. The resulting execution is summarized in the Figures 10 and 12 while the corresponding perspective of the operator control station which contains the 3D world model and footstep plan is showed in the Figures 9 and 11. We also provide videos of stepping over the pitch and the chevron hurdle at `https://www.youtube.com/watch?v=7Qv__bLa3j4` and `https://www.youtube.com/watch?v=vAtqVKGWvFM`.

In general we are able to generate plans on flat surfaces within a few seconds using a single core of a Core i7 computer. Going into 3D space, the initial solution is found within a few seconds too, but the improvement of the plan takes around ten seconds depending on terrain complexity. The terrain model generation time depends on the selected size of the region of interest; but at the DRC Trials it took a maximum of one second for a region of the size which is visible in Figures 9 and 11. The plan for the pitch ramp in Figure 9 could be generated online in 0.7 seconds and for the chevron hurdle in Figure 11 in 1.8 seconds. To our knowledge almost all participating teams haven't performed the terrain task by using a high-level footstep planner. Instead each step was defined by the operator manually which takes obviously more planning time.

As discussed in Section IV the footstep planner

is also required to navigate through narrow doorways. The video at https://www.youtube.com/watch?v=BlUfl5iSAkU shows the successful attempt of the robot walking autonomously through a very small doorway without any collisions using our footstep planner.

VIII. Conclusion

In this work a complete supervised 3D footstep planning system covering perception, world modeling, full 3D (6 DoF) planning and operator interaction was introduced. We demonstrated a graph-based footstep planning approach utilizing an existing black box walking controller to generate whole sequences of steps in rough terrain scenarios. The 3D state of each step is determined by the planner automatically. Thus, the operator has not to define manually each step to be executed which increases the operator efficiency and mission performance.

Many features like the used full step representation with reachability polygon as well as the novel ground contact estimation based on height map and normal estimation improve the planner performance in uneven terrain significantly. The generated terrain model provides sufficient data to perform 3D footstep planning in unstructured environments and a compressed version can be sent to the operator station to provide situational awareness. Furthermore, we described a couple of user interfaces that were implemented to provide flexible footstep planning control given in different abstraction layers.

The first simulation-based experiments show a proof of concept for learning cost functions for black box walking controller. However, it turned out that the investigated offline learning approaches are not feasible for robots which are still in development due to required excessive training.

In future work we would like to improve planning results by generating the reachability polygon (see Section V-A.2) online by using inverse kinematics and respecting in parallel the center of mass dynamics including all carried objects. Experience has shown us that the terrain model is crucial for reliable 3D footstep planning. For this reason we will investigate how to improve the model quality, while reducing the computational expense. The effectiveness of the supervising operator depends on the provided options to interact with the footstep planner. Therefore we are going to implement an interactive footstep planner interface which will allow the operator to fine tune the planned steps. The advanced planner will also incorporate plan stitching to allow more complex and flexible "human-in-the-loop" planning which mitigates the lack of intelligence of the planning system.

IX. Acknowledgment

The authors would like to thank all members of Team ViGIR for their contribution and support which enabled the realization of this work. This work has been funded by DFG Research Training Group 1362 and Defense Advanced Research Projects Agency (DARPA) under Air Force Research Lab (AFRL) contract FA8750-12-C-0337; the views expressed in this paper are those of the authors.

References

[1] Kohlbrecher, S., Romay, A., Stumpf, A., Gupta, A., von Stryk, O., Bacim, F., Bowman, D., Goins, A., Balasubramanian, R., Conner, D.: Human-robot teaming for rescue missions: Team ViGIR's approach to the 2013 DARPA Robotics Challenge Trials. Journal of Field Robotics (to appear)

[2] Hirukawa, H., Hattori, S., Kajita, S., Harada, K., Kaneko, K., Kanehiro, F., Morisawa, M., Nakaoka, S.: A pattern generator of humanoid robots walking on a rough terrain. In: IEEE Intl. Conf. on Robotics and Automation. (2007) 2181–2187

[3] Takubo, T., Imada, Y., Ohara, K., Mae, Y., Arai, T.: Rough terrain walking for bipedal robot by using ZMP criteria map. In: IEEE Intl. Conf. on Robotics and Automation. (2009) 788–793

[4] Ratliff, N., Zucker, M., Bagnell, J.A., Srinivasa, S.: CHOMP: Gradient optimization techniques for efficient motion planning. In: IEEE Intl. Conf. on Robotics and Automation. (2009) 489–494

[5] Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., Abbeel, P.: Finding locally optimal, collision-free trajectories with sequential convex optimization. In: Robotics: Science and Systems. (2013)

[6] Bouyarmane, K., Kheddar, A.: Multi-contact stances planning for multiple agents. In: IEEE ICRA. (2011) 5246–5253

[7] Bouyarmane, K., Vaillant, J., Keith, F., Kheddar, A.: Exploring humanoid robots locomotion capabilities in virtual disaster response scenarios. In: IEEE-RAS HUMANOIDS. (2012) 337–342

[8] Chestnutt, J., Lau, M., Cheung, G., Kuffner, J., Hodgins, J., Kanade, T.: Footstep planning for the honda ASIMO humanoid. In: IEEE Intl. Conf. on Robotics and Automation. (2005) 629–634

[9] Garimort, J., Hornung, A.: Humanoid navigation with dynamic footstep plans. In: IEEE ICRA. (2011) 3982–3987

[10] Hornung, A., Dornbush, A., Likhachev, M., Bennewitz, M.: Anytime search-based footstep planning with suboptimality bounds. In: IEEE-RAS Intl. Conf. on Humanoid Robots. (2012) 674–679

[11] Hornung, A., Maier, D., Bennewitz, M.: Search-based footstep planning. In: ICRA Workshop Progress & Open Problems in Motion Planning & Navigation for Humanoids, Karlsruhe, Germany. (2013)

[12] Maier, D., Lutz, C., Bennewitz, M.: Integrated perception, mapping, and footstep planning for humanoid navigation among 3d obstacles. In: IEEE/RSJ IROS. (2013) 2658–2664

[13] Kalakrishnan, M., Buchli, J., Pastor, P., Schaal, S.: Learning locomotion over rough terrain using terrain templates. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems. (2009) 167–172

[14] Kolter, J.Z., Rodgers, M.P., Ng, A.Y.: A control architecture for quadruped locomotion over rough terrain. In: IEEE Intl. Conf. on Robotics and Automation. (2008) 811–818

[15] Parlaktuna, O., Ozkan, M.: Adaptive control of free-floating space manipulators using dynamically equivalent manipulator model. Robotics and Autonomous Systems **46**(3) (2004) 185–193

[16] Zucker, M., Bagnell, J.A., Atkeson, C.G., Kuffner, J.: An optimization approach to rough terrain locomotion. In: IEEE Intl. Conf. on Robotics and Automation. (2010) 3589–3595

[17] Deits, R., Tedrake, R.: Footstep planning on uneven terrain with mixed-integer convex optimization. In: IEEE-RAS Intl. Conf. on Humanoid Robots. (2014) to appear

[18] Chilian, A., Hirschmuller, H.: Stereo camera based navigation of mobile robots on rough terrain. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems. (2009) 4571–4576

[19] Nelson, G., Saunders, A., Neville, N., Swilling, B., Bondaryk, J., Billings, D., Lee, C., Playter, R., Raibert, M.: Petman: A humanoid robot for testing chemical protective clothing. Advanced Robotics **30**(4) (2012) 372–377

[20] Wurm, K.M., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In: ICRA Workshop best practice in 3D perception & modeling for mobile manipulation. Volume 2. (2010)

[21] Likhachev, M., Gordon, G.J., Thrun, S.: ARA*: Anytime a* with provable bounds on sub-optimality. In: NIPS. (2003)

[22] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Pearson Ed. (1994)

[23] Sprunk, C., Lau, B., Pfaffz, P., Burgard, W.: Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In: IEEE Intl. Conf. on Robotics and Automation. (2011) 72–77

[24] Romay, A., Kohlbrecher, S., Stumpf, A., von Stryk, O.: Template-based manipulation for supervised semi-autonomous humanoid robots. In: IEEE-RAS Intl. Conf. on Humanoid Robots. (2014) to appear