This is a preprint of a paper that appeared in the Proceedings of
IEEE-RAS International Conference on Humanoid Robots, pp. 938-945, Nov 15-17, 2016

Open Source Integrated 3D Footstep Planning Framework for Humanoid Robots

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

***Abstract*— Humanoid robots benefit from their anthropomorphic shape when operating in human-made environments. In order to achieve human-like capabilities, robots must be able to perceive, understand and interact with the surrounding world. Humanoid locomotion in uneven terrain is a challenging task as it requires sophisticated world model generation, motion planning and control algorithms and their integration. In recent years, much progress in world modeling and motion control has been achieved. This paper presents one of the very first open source frameworks for full 3D footstep planning available for ROS which integrates perception and locomotion systems of humanoid bipedal robots. The framework is designed to be used for different type of humanoid robots having different perception and locomotion capabilities with minimal implementation effort. In order to integrate with almost any humanoid walking controller, the system can easily be extended with additional functionality that may be needed by low-level motion algorithms. It also considers sophisticated human-robot interaction that enables to direct the planner to generate improved solutions, provides monitoring data to the operator and debugging feedback for developers. The provided software package consists of three major blocks that address world generation, planning and interfacing low-level motion algorithms. The framework has been successfully applied to four different full-size humanoid robots.**

I. Introduction

In the last decades significant progress has been achieved in the field of humanoid robots. However, the capabilities of today's humanoid robots are still far from meeting the requirements of real world applications such as disaster response or reliable assistance at work or in the home. As the complexity of real world scenarios goes beyond what autonomous humanoid robots may be able to solve in the future, a remote human supervisor should be able to assist the humanoid robot in perception and planning to leverage human intelligence and cognitive abilities whenever needed.

The control of robots should not overburden human operators. Consequently, the interdependence of human operators and humanoid robots must be considered for any complex system to enable appropriate human-robot-interaction where each member of the human-robot team can contribute to the mission when required [1].

Human-made environments are filled with ground level changes, door thresholds, stairs, etc. Therefore, a crucial capability for the use of humanoids is the ability to traverse inclined terrain and narrow corridors in possibly harsh environments with minimal risk of failure.

*Department of Computer Science, Technische Universität Darmstadt
stumpf,kohlbrecher,stryk@sim.tu-darmstadt.de
†Christopher Newport University david.conner@cnu.edu

This paper presents a new open source footstep planning framework (Sect. III) which integrates environment perception (Sect. III-G) and is usable by different types of humanoid robots with only minimal required adjustments. It adopts state of the art search-based planning algorithms (e.g. ARA*) and provides crucial methods and tools for automatic generation of feasible sequences of full 3D footstep placements over uneven terrain among obstacles. Different levels of interactivity (Sect. IV) allow the human supervisor to direct the planner how to solve the current locomotion planning task properly. In the other direction, the footstep planning system assists the user to keep all footstep placements valid and safe to execute for the robot.

Apart from the difficulties in human robot interaction, the development of a (semi-)autonomous humanoid robot system itself is already a tremendous challenge. Modularity and reusability of basic building blocks for the highly complex humanoid robot capabilities are key to advancements in research and development. Therefore, the existing footstep planner [2] has been developed into an open source framework that enables uncompromising and seamless integration with many humanoid robot systems as presented in Sect. V. This demanding goal is achieved by unique features such as a comprehensive plugin management (Sect. III-B) and parameter system (Sect. III-C). These tools provide the required flexibility to adapt to almost any bipedal humanoid robot with few lines of code. Additional toolboxes can handle common tasks such as world model generation, step queue management (Sect. III-F) or interfacing low-level motion algorithms. A core strength of our approach is not only to adapt planning policies but step data representation as well. The presented work can be either used as a 3D planner out of the box, as research tool in bipedal search-based planning or for benchmarking walking controllers in difficult terrain. It also allows for a number of extensions (Sect. V-E).

The implementation of the presented footstep planning framework is based on an extension of Team ViGIR's approach to the DRC [3][4] and is available on GitHub open source. Full documentation and tutorials can be found at the ROS Wiki[1].

II. Related Work

The number of available full-size humanoid robots is steadily increasing with significant progress towards bipedal locomotion. Improved control policies allow for more autonomous navigation, in particular for traversing rough ter-

[1]http://wiki.ros.org/vigir_footstep_planning

rain. Although footstep planning has been investigated intensively in the last years, only few approaches consider footstep height [5] and even fewer consider full six degree of freedoms (DoF) foot placements [6]. While many interesting results have been achieved, there is still a lack of publications of reusable and extensible open source implementations of such approaches.

In terms of migration effort, the most portable approaches for footstep planning are contact-before-motion methods that decouple the planning and motion layer from each other. Prior work has investigated various contact-before-motion approaches such as whole-body-planning [7][8][9], walk pattern generation [10][11], convex optimization [6][12] or dynamic programming [13][14][15].

Many of the previously published research approaches to humanoid locomotion planning are, to the best of our best knowledge, not available as software in public. However, there are a few software solutions available open source.

Coleman et. al. present a proof of concept for motion planning for HRP2 as extension for the MoveIt! framework[2]. The prototype version is able to generate whole-body-motions plans for HRP2 which includes footstep planning as well, but requires on the other side a whole-body-motion controller to execute them. Therefore this approach is not well suited for pure walking controllers as the planned whole-body-motions may likely not be executed properly.

Recently the Humanoid Path Planner (HPP) [16] was released. The authors describe HPP as an answer to the lack of a standard framework for complex motion planning problems, such as constrained navigation among movable objects. In general, the authors claim that HPP is easy to use and extendable by overloading abstract classes. Although HPP solves a large set of humanoid motion planning tasks, it addresses neither operator interaction nor perception.

In general only few planning frameworks for humanoid locomotion are available and therefore many human resources are wasted in implementation of humanoid navigation systems. This motivates our open source footstep planning framework that can be deployed quickly with any humanoid system especially for ROS [17]. We intend to establish a community similar to MoveIt![3] in manipulation and ROSControl[4] in controller domain already do.

Fortunately Hornung et. al. have already implemented an open source 2D footstep planner [18] for ROS. It is very easy to use due to standardized ROS messages, but provides limited adaptability. In order to consider robot specific constraints, the base code as well as messages have to be changed directly such as we did in previous work [2]. We noticed the lack of a versatile solution that causes a huge implementation overhead when migrating the software to other robot platforms. This inconvenient situation motivated our follow-up work presented in this paper.

[2]https://github.com/davetcoleman/moveit_hrp2
[3]http://moveit.ros.org
[4]http://wiki.ros.org/ros_control

III. Footstep Planning Framework

A. Introduction

A software framework is a generic abstraction of particular functionality as part of a larger software platform. The framework provides sufficient options to be extended to an application-specific software by changing functionality with additional user-written code. Thus, our main objective is to provide a capable footstep planning framework that integrates environmental perception and may be deployed easily into an existing ROS setup. For this, the planner has to provide a complete set of basic functionalities, but also has to be extendable for new application-specific features (like alternative planning policies or robot specific footstep data). Based on this structure, any new user of the framework can build upon the already implemented and proven algorithms and needs to focus only on implementation of robot specific elements. This decreases the likelihood of errors and greatly reduces development time by saving a lot of implementation effort.

The folllowing paragraphs describe the basic blocks in order to achieve this goal by implementing a parameter[5] and plugin[6] management system. Although these packages are introduced in the context of footstep planning, they can be freely used in any other project as well.

B. Plugin Management

The *vigir_pluginlib* is the key asset to manage heterogeneous types of plugins. Our solution is based on *pluginlib*[7] which already provides the basic capability to load classes (plugins) from shared libraries via user-written code by using the ROS build infrastructure.

Plugins are the basic units to inject efficiently user specific code into a framework while all existing code is preserved. The idea of using plugins as small code packets follows the divide and conquer software paradigm. Each plugin solves a problem partially, but they can be composed to a plugin set forming a solution of a complex problem.

The plugin manager can easily determine the provided functionality of a plugin by checking its inheritance tree. This ability can be used by the developer to gather all relevant plugins from a heterogeneous plugin database needed for a specific part of the framework or algorithm (a generic example is given in Fig. 1). This allows the use of any kind of plugin in transparent manner, whereby implementation details are hidden. The manager itself sets up all ROS services and action servers automatically that facilitate generic access to all management functionality as used by the accompanied graphical user interface.

C. Parameter Management

In real world applications, different terrain scenarios have to be addressed (e.g. flat surface, narrow floors, stairs or sloped terrain). The footstep planner can perform best if a

[5]http://wiki.ros.org/vigir_generic_params
[6]http://wiki.ros.org/vigir_pluginlib
[7]http://wiki.ros.org/pluginlib

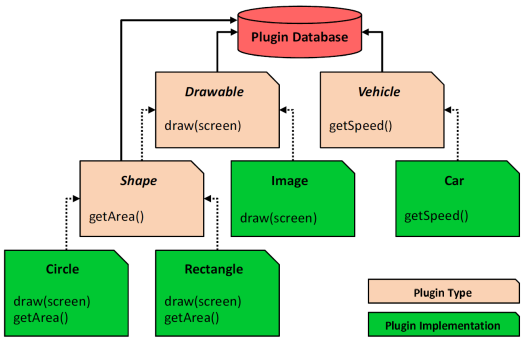


Fig. 1: Generic example inheritance tree for plugins in a heterogeneous plugin database. When the user requests "***Drawable***" plugins, the plugin manager will automatically resolve this request by returning all instantiated "**Circle**", "**Rectangle**" and "**Image**" plugins.

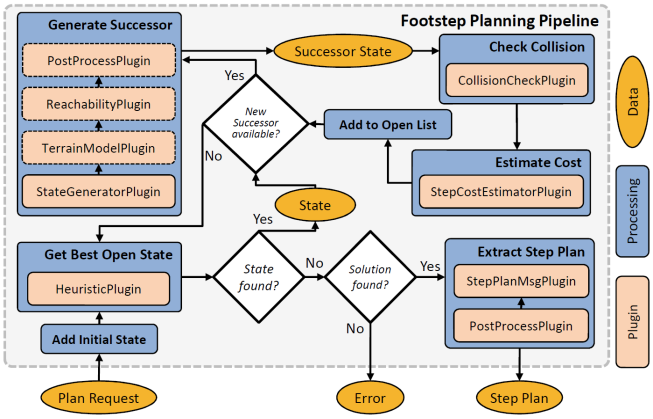


Fig. 2: Plugins embedded into footstep planning pipeline.

dedicated parameter set has been defined for each kind of scenario.

The *vigir_generic_params* solves the conflicting issue of having a fixed ROS message structure at build time and the varying content of parameter sets due to individual compositions of plugins during run time. It is designed as substitution for the classical ROS Parameter Server while using the same YAML syntax for configuration files and providing the same interfaces for accessing parameters. This design consideration makes it very convenient to use for ROS-familiar developers. The *vigir_generic_params* implements additional features such as a flexible structure for parameter sets that can be shared via ROS messages and a parameter set management to allow quick software reconfiguration by just providing multiple configuration files. In addition a generic graphical user interface allows modifying those parameter sets online.

D. Footstep Planning Library

The parameter and plugin system delivers the adaptability needed to cope with different humanoid systems and therefore form the basic infrastructure of the proposed footstep planning framework. In order to apply the plugin system, the footstep planner pipeline has been investigated for injection points where a user might want to affect the behavior of the planner. For each such injection point, a plugin type has been defined:

- *CollisionCheckPlugin*: Collision checks of a given state and/or transition.
- *HeuristicPlugin*: Computes heuristic value (estimated remaining cost) to the goal state.
- *PostProcessPlugin*: Post-processing of a generated single step or step plan.
- *ReachabilityPlugin*: Defines the set of valid step transitions.
- *StateGeneratorPlugin*: Determines the next state(s) to visit.
- *StepCostEstimatorPlugin*: Estimates cost and risk for given step transition.
- *StepPlanMsgPlugin* (singleton): Marshalling interface for robot specific data carried in each single step and step plan.
- *TerrainModelPlugin* (singleton): Provides 3D terrain model of environment.

The last two plugin types are defined as singleton, thus the plugin manager allows only a single instance of this plugin type to be loaded while other plugin types may exist multiple times or even as different implementations in the plugin database. Each plugin of the same type contributes individual computational results, e.g. the results of all active *StepCostEstimatorPlugins* are summed up to a total step cost. Fig. 2 shows when each plugin type takes effect on the planner pipeline. For a quick deployment of the framework, plugin implementations for common cases already exist for mandatory plugin types. Furthermore plugins do not live in isolation, which should be taken into account because they can also access and use each other as well. The most common case is the *StepPlanMsgPlugin* that allows custom plugins to access robot specific variables encoded into the step plan.

One of our main goals is keeping the overhead of the plugin system low in order to maintain high planning efficiency. It is clearly inefficient to request needed plugins from the manager for each single iteration during planning. For this reason, the planner retrieves all plugins once at the beginning of each planning request and initializes them with the given parameters for this request.

E. Framework Extension

Next, we provide an example how to extend our framework, demonstrating that only a few steps have to be performed.

In order to refine the step cost estimation for each single step, the developer has first to add a new implementation of the plugin type *StepCostEstimatorPlugin* named e.g. "*my_step_cost_estimator*" according to the instruction at the related ROS Wiki[8]. Afterwards this new plugin and its containing code can be used immediately by the planner by

[8]http://wiki.ros.org/vigir_pluginlib

modifying the used plugin set. It also allows the user to assign values to parameters used by this plugin. An example plugin set configuration file follows:

```
plugin_sets:
  default:
    [...]

    # StateGeneratorPlugin
    reachability_state_generator: none

    # StepPlanMsgPlugin
    step_plan_msg_plugin: none

    # ReachabilityPlugins
    reachability_polygon: none

    # StepCostEstimatorPlugins
    const_step_cost_estimator: none
    euclidean_step_cost_estimator: none
    my_step_cost_estimator:
      params:
        my_param: 42

    # HeuristicPlugins
    step_cost_heuristic: none
    euclidean_heuristic: none

    [...]
```

It is notable that changing the plugin set does not require recompilation of the software. A modification of the plugin set can also be performed while the software is running, allowing for highly dynamic behavior changes when needed. This example demonstrates how the plugin system allows to add individual code into any stage of planning with minimal effort.

F. Step Controller

While working on different humanoid robots, the authors noticed that many low-level controllers provide very capable but different interfaces that enable complex locomotion execution such as continuous walking over rough terrain. In order to use the walking controller most effectively, step queue management is required that allows the system to update already queued steps as well as add new steps to the step plan.

For this reason we developed a step controller tool[9] for our footstep planning library which is designed to directly interface with a low-level walking controller. The basic implementation consists of a state machine that flushes single steps to the walking controller when required and a step queue which is capable of seamless integration of step plan updates like step plan stitching. Hereby, step plan stitching denotes a method to transform two step plans according to a common pivot step, thus they can be merged cleanly. The implementation of the step controller follows the same paradigms as the footstep planner. The behavior can thus be adapted by implementing a robot specific *StepControllerPlugin*.

[9]http://wiki.ros.org/vigir_step_control

For human supervision, the step controller also frequently reports the internal state machine and step execution feedback that is also visualized by the accompanied user interface.

G. Terrain Modeling

3D footstep planning requires terrain height and surface normal data of the surrounding environment that must be efficiently accessible in order to enable short planning times. Although there is already preceding work such as OctoMap [19] available, these approaches are focused on mapping or surface reconstruction. OctoMap is a very capable probabilistic mapping system, but requires a lot of computational resources for high resolution grid maps that are crucial for rough terrain tasks. In addition the included surface normal estimation is based on marching cubes, lacking in precision.

Therefore, we decided to implement a light-weight approach allowing to generate the required terrain data for 3D footstep planning online. In order to enable efficient data access during planning we split the terrain model into two parts: A grid-based elevation map and a 3D octree-based data structure storing all captured point cloud data and the associated estimated normals. Updating both data structures can be performed efficiently and enables a very efficient two level lookup to determine the full 3D pose of each foot placement: For each 2D position the corresponding z-value is first looked up in the elevation map and afterwards used to obtain the surface orientation stored in the octree.

In our previous work [2] we already chose PCA in combination with a kd-tree-based data structure to estimate surface normals that is according to Klasing et. al. [20] the best approach for quick and accurate normal surface estimation using noisy point clouds. The computation is speed up by cropping each laser scan by the region of interest before updating the elevation map and surface normals. Using scans from a spinning laser scanner, this procedure can be performed in real-time as demonstrated in Fig. 3. In general, this approach enables terrain model generation on any robot system that provides point clouds given in global reference frame.

IV. Operator Interaction

For supervised robot operation, different abstraction layers for accessing the footstep planner capabilities are available. Although the planner is able to generate feasible plans, in practical application, there always remains a possibility that the resulting plan contains undesirable steps due to noisy sensor data. For this reason, the footstep planning system was extended to provide multiple services to manage footstep plans in an interactive and coactive manner [1] allowing for human intelligence to assist in perception and planning. For instance, these services enable the human supervisor to assist the planner by quickly adjusting single steps while the planner automatically aligns the 3D foot pose with respect to the underlying surface orientation and reports back immediately if the modified step sequence is still safe to execute. In a more high level manner, the operator can

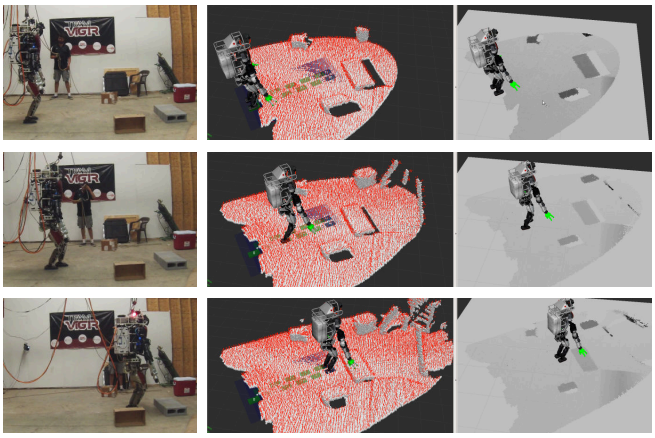


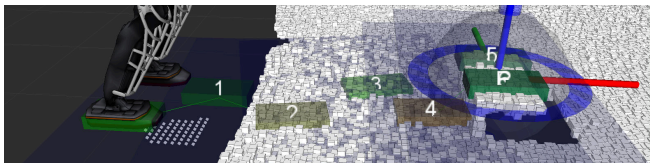
Fig. 3: Online terrain generation while robot walks among obstacles (left). In the middle the aggregated point cloud with surface normal estimate (red) is viusalized. The generated height map is shown on the right. The corresponding video can be watched online in full length at *http://youtu.be/S-hhHFB77Co*.

also build up complex sequences of footsteps by stitching multiple smaller step plans together. The interactive planning mode significantly improves mission performance during locomotion tasks, which is demonstrated in Fig. 4. A video presentation of interactive planning is available under *http://youtu.be/kX4rNbo5UYk* which illustrates how the operator is able to direct the planner to improve the footstep plan. An example of the planner's assistance capability can be seen at the 0:26 mark in the video when the footstep planner clearly notifies the operator that he moved the step too far away from the successor step which would result in unsafe footstep execution.

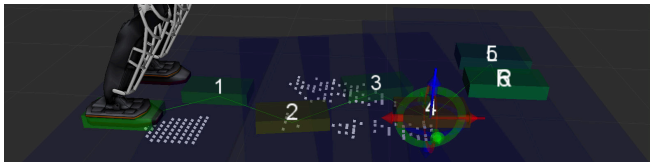
When working with robots, situational awareness becomes crucial for an operator. The operator must be aware of the environment as well as the internal state of the robot. In scenarios with many obstacles or difficult terrain, the planner will take more time than usual to deliver suitable results. In this case it is desirable to get continuous feedback from the planner to ensure that the planning process is still working properly. Fig. 5 demonstrates how our footstep planning system frequently reports the current planning status back to the operator. The operator is actually able to comprehend the current planning progress and even able to detect if the planner is stuck or struggles at certain obstacles. Then the operator is able to preempt the current planning process anytime and direct the planner by changing parameters or improving planning policies. Further debug options are available by visualizing the state expansions of the used ARA* algorithm (see Fig. 5) which helps to shape better planning policies.

V. Results

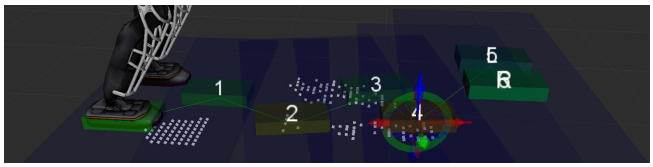
The requirements for the footstep planning framework are minimal and all advanced features can be accessed incrementally. For footstep planning and execution on flat



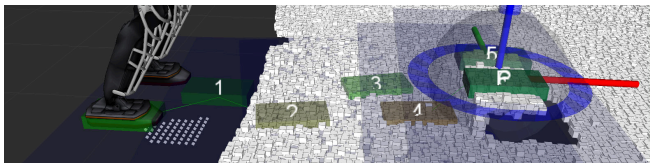
(a) The operator is not satisfied with placement of step 4 as it is too close in front of the cinder block.



(b) Operator selects step 4 for editing. Terrain model has been hidden for a better visibility of interactive marker.



(c) Step 4 has been slightly moved away from the cinder block by the operator.



(d) Final result of modified footstep plan which is ready for execution now.

Fig. 4: Example of interactive footstep planning where the operator modifies a single step of a pre-generated plan. The steps are automatically colored by the planner in a range from green to red to indicate risk level of execution.

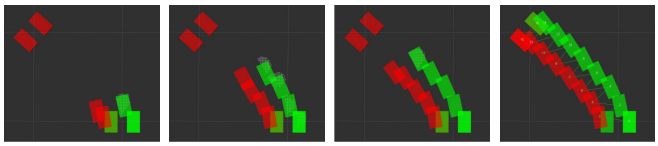


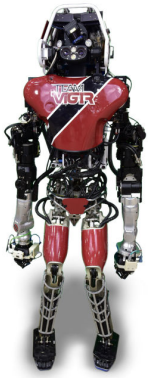
Fig. 5: Realtime feedback reported frequently by the planner shows the current planning progress. White dots show recently expanded states by the A* algorithm.

surfaces only a walking controller suitable for flat terrain is needed. Collision avoidance can be considered as soon an occupancy grid map is provided to the footstep planner. For uneven terrain the robot has to provide incremental 3D point cloud data given in a global reference frame for the terrain model generator and requires a suitable walking controller to perform the individual steps needed to cross the terrain. In all cases the planning system assumes that the walking algorithm is requesting foot placements as input.

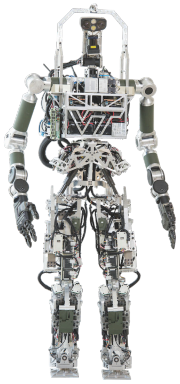
The presented entire software stack is fully ROS-enabled to facilitate the usage for the ROS community and is available at GitHub open source.

A. Robots Using The Framework and Software

The presented software has currently been deployed on four different robots, each delivered with its own walking control software: Team ViGIR's Atlas, Team VALOR's ESCHER and Team Hector's THORMANG 1 and 3 (Fig. 6). In the case of Atlas, we interfaced BDI as well as IHMC control software [21] which sums up to five different walking controllers that have been interfaced successfully so far.

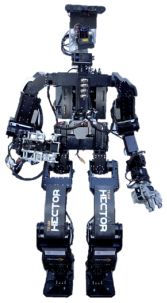


(a) Atlas (DRC Finals)



(b) ESCHER

(c) Johnny #5 (DRC Finals)



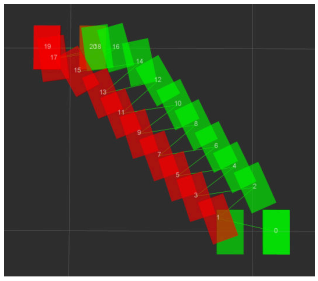
(d) Johnny #5 (2016)

Fig. 6: Robots used for evaluation.

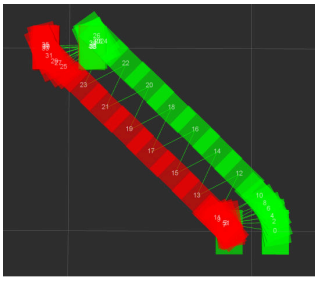
Atlas is the only hydraulically driven humanoid robot which requires different control approaches compared to electrically driven ones. The resulting varied walking capabilities and constraints motivate the implementation of an adaptable footstep planning system that can handle each robot individually. This applies for footstep data representation as well as footstep planning policies. For instance the 3D foot trajectory generation of the underlying walk algorithms require footstep data in different formats. BDI's walking algorithm demands only for a lift height and a swing height while IHMC's approach accepts intermediate travel points for the foot as well as the convex hull of expected ground contact. Both scenarios could be handled by our framework seamlessly. An example for different planning policies is given in the following section.

B. Portability

The plugin system enables the adaptability needed to support different humanoid robots and allows for quick prototyping as well as highly dynamic behavior modifications due to different code that can be executed dynamically. When only using parameters, this behavior can only be achieved by enormous implementation effort. As an example, the planning policies can be quickly changed by swapping plugins as demonstrated in Fig. 7. The usual omnidirectional walk (see Fig. 7a) can be easily constrained to a rotate-translate-rotate (RTR) movement (see Fig. 7b) which was required by the first generation Johnny #5 robot due to limited walking capabilities. Although the RTR plan looks smoother than the omnidirectional plan, it uses a higher number of steps, resulting in longer travel time. This drawback is mitigated by using the non-constrained planner generating omnidirectional plans.



(a) Omnidirectional movement



(b) RTR movement

Fig. 7: Different planning policies can be achieved by just swapping plugins. Here, the omnidirectional planner (a) has been constrained to RTR movements (b).

Another benefit of the plugin system is interchangeability. In general, approaches or algorithms implemented as plugins can be used in different projects as long they do not depend on robot specific sources. This easily allows everyone to share and contribute their own implementations across the community and reduces the need for reimplementation. This paradigm encourages code stability as existing code remains unchanged when new plugins are added to the system. Therefore, the footstep planning system provides a tool for research in search-based planning policies especially for difficult terrain scenarios.

Our contribution to Team VALOR shows that one of our main goals have been clearly achieved. We have been able to contribute high level perception and footstep planning software to another team whose research focus has been more on other aspects like design and control. The deployment of our footstep planning system enabled them to benchmark their control approach in humanoid locomotion on real terrain quicker and even to use it in the DRC Finals. For more details we refer to [22]. This case study shows how our footstep planning framework can speed up evaluation and usage of robot systems, especially when projects are focused on actuation and control design.

As already introduced Johnny #5 is currently based on 3rd generation of THORMANG and is supplied by ROBOTIS with a walking engine. We have recently implemented a basic integration to the newly released ROBOTIS framework

that is available at GitHub[10,11]. The code shows the minimal effort spent to deploy the footstep planning system in a new robot system. In this particular case, only robot specific configuration files and the *ThorMangStepPlanMsgPlugin* which is based on *StepPlanMsgPlugin* is needed to translate robot specific data to get our framework running properly. Furthermore, we also provide a plugin for the step controller[12] which enables the full potential of the step controller while providing only robot specific code. Latest results can be seen in the video recorded from the Humanoids@Rescue demonstration at RoboCup 2016[13]. Here, all walked steps have been planned by the presented footstep planner and executed via the step controller. Especially at the 1:17 mark the robot was successfully able to follow a large sequence of footstep placement generated by our footstep planner.

In the following section we demonstrate the high capability of our planning framework with Atlas as it had the best locomotion capabilities available to us until now.

C. Perception

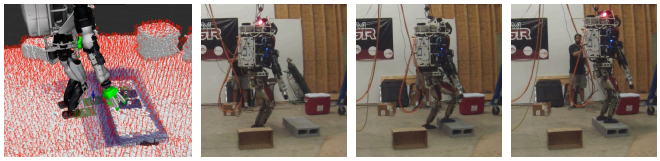


Fig. 8: Atlas stepping up successfully on a cinder block based on previously online generated terrain data (see Fig. 3).

Online terrain generation is worthwhile because then up-to-date terrain data is always available, even when the robot is walking. Depending on the accuracy of the robot state estimation, the robot is able to continue walking over rough terrain without waiting to gather sufficient terrain data. In case of Atlas the accuracy condition is met very well and after a short walk in front of the cinder block (see Fig. 3) the robot is immediately able to step up on it flawlessly (see Fig. 8).

D. Planning

While the implementation of the previous footstep planner [2] experienced heavy refactoring into the new framework, the planning efficiency has always been considered. As explained in Section III-D, the overhead of the plugin system has been reduced to a minimum by design. On the other hand, all unique features from the first footstep planner version such as ground contact estimation for overhanging steps have been preserved in the new framework.

To evaluate planning time we set up the pitch ramp and chevron hurdle from DRC Trials. We ran tests isolated (Fig. 9+10) and combined (Fig. 11).

[10]https://github.com/thor-mang/thor_mang_footstep_planning_plugins

[11]https://github.com/thor-mang/thor_mang_footstep_planner

[12]https://github.com/thor-mang/ROBOTIS-THORMANG-MPC/blob/indigo-devel/thormang3_step_control_module/src/robotis_online_walking_plugin.cpp

[13]http://youtu.be/uoyYznea0Pk

Fig. 9: Step plan crossing the pitch ramp.

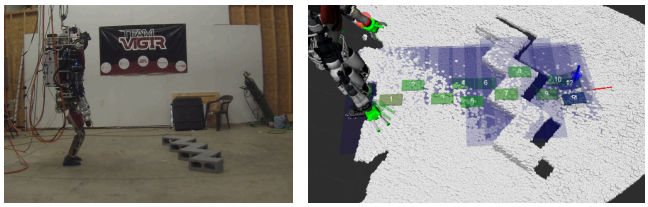


Fig. 10: Step plan crossing the chevron hurdle.

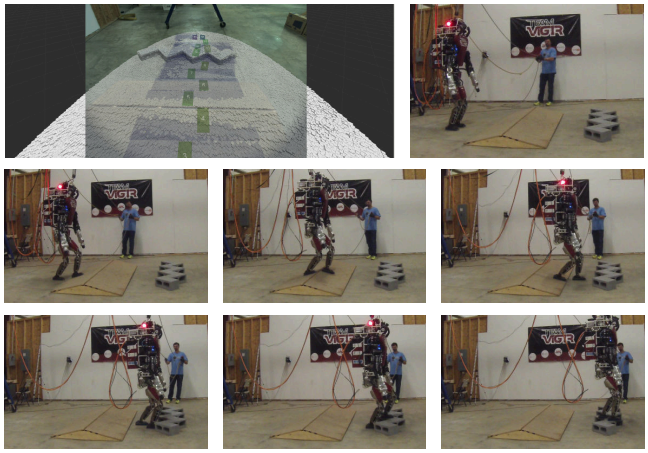


Fig. 11: Atlas walks non-stop over mixed terrain following the footstep placements generated by our footstep planning framework. The upper left image shows the terrain model and planned footstep placements projected into the robot's camera view. The full video is available online at *http://youtu.be/Fd2D-XG9VOg*.

In all tests the footstep planner is able to generate suitable sequences of footstep placement in short amount of time (see TABLE I). Although the numbers show that the new planner is slower, here the travel distance must be taken into account. The goal distances are almost doubled compared to the DRC Trials plans. Considering that planning complexity rises exponentially with the distance to the goal position, this is still a good and comparable result. The combined test must be highlighted here as in spite of increased terrain complexity the planner is able to generate a solution almost in the same time as only the chevron hurdles takes. In every test case the Atlas robot is also able to cross the obstacles by executing the planned steps as shown in Fig. 11.

E. Advanced Extensions

Although the presented work is focused on pure footstep placement generation, the framework allows to extend the

Terrain	DRC Trials[2]		Post DRC Finals	
Pitch Ramp	0.7s	8 steps	3.9s	10 steps
Chevron Hurdle	1.8s	8 steps	4.3s	13 steps
Mixed	N/A	N/A	4.4s	14 steps

TABLE I: Planning times based on experiments illustrated in Figures 9-11.

approach to more complex scenarios such as multi-contact planning. For this purpose a *StateGeneratorPlugin* can extend the currently used state by e.g. hand poses. In order to consider the full new state during planning, the new state generator has at least to be accompanied by proper cost, heuristic, collision checking plugins. These plugins can also interface and use external whole body frameworks.

VI. Conclusion

In this work, an open source footstep planning framework is described. It integrates perception, world modeling, full 3D planning, step execution tracking, human supervision and coactive planning, while being modular and extensible.

A versatile plugin library system is presented that allows the user to add new code into the planning framework with few line of code. This powerful tool enables to provide a modular footstep planning library that can be adapted to almost any humanoid robot. Furthermore, the footstep planning library allows for better collaboration between humanoid locomotion projects due to the interchangeability of plugins and standardized interface that simplifies sharing of algorithms and code. We demonstrated how the presented work can be either used as a 3D planner, as research tool, or for benchmarking walking controllers.

In future work, we would like to extend the standard plugin library by state of the art methods as well as novel approaches. We are also working on getting the interactive planning capability available for RViz which grants more convenient access to the high level coactive planning functionality out of the box.

VII. Acknowledgment

The authors would like to thank all members of Team ViGIR and Team Hector for their contribution and support which enabled the realization of this work. This work has been funded by 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] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, B. Van Riemsdijk, and M. Sierhuis, "The fundamental principle of coactive design: Interdependence must shape autonomy," in *Coordination, organizations, institutions, and norms in agent systems VI*. Springer, 2011, pp. 172–191.

[2] A. Stumpf, S. Kohlbrecher, D. C. Conner, and O. von Stryk, "Supervised footstep planning for humanoid robots in rough terrain tasks using a black box walking controller," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014, pp. 287–294.

[3] S. Kohlbrecher, A. Romay, A. Stumpf, A. Gupta, O. Von Stryk, F. Bacim, D. A. Bowman, A. Goins, R. Balasubramanian, and D. C. Conner, "Human-robot teaming for rescue missions: Team ViGIR's approach to the 2013 DARPA Robotics Challenge Trials," *Journal of Field Robotics*, vol. 32, no. 3, pp. 352–377, 2015.

[4] S. Kohlbrecher, A. Stumpf, A. Romay, P. Schillinger, O. Von Stryk, and D. C. Conner, "A comprehensive software framework for complex locomotion and manipulation tasks applicable to different types of humanoid robots," *Frontiers in Robotics and AI*, vol. 3, p. 31, 2016.

[5] D. Maier, C. Lutz, and M. Bennewitz, "Integrated perception, mapping, and footstep planning for humanoid navigation among 3d obstacles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 2658–2664.

[6] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014, pp. 279–286.

[7] K. Bouyarmane and A. Kheddar, "Multi-contact stances planning for multiple agents," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5246–5253.

[8] K. Bouyarmane, J. Vaillant, F. Keith, and A. Kheddar, "Exploring humanoid robots locomotion capabilities in virtual disaster response scenarios," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 337–342.

[9] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[10] K. Harada, S. Hattori, H. Hirukawa, M. Morisawa, S. Kajita, and E. Yoshida, "Motion planning for walking pattern generation of humanoid," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 4227–4233.

[11] 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 (IROS)*, 2015, pp. 2780–2786.

[12] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 881–888.

[13] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2003.

[14] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda ASIMO humanoid," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 629–634.

[15] A. Hornung, D. Maier, and M. Bennewitz, "Search-based footstep planning," in *ICRA Workshop Progress & Open Problems in Motion Planning & Navigation for Humanoids*. IEEE, 2013.

[16] J. Mirabel, S. Tonneau, P. Fernbach, A.-K. Seppala, and M. Campana, "HPP: A new software for constrained motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*. IEEE, 2009.

[18] J. Garimort and A. Hornung, "Humanoid navigation with dynamic footstep plans," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3982–3987.

[19] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[20] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3206–3211.

[21] T. Koolen, S. Bertrand, G. Thomas, T. De Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a momentum-based control framework and application to the humanoid robot Atlas," *International Journal of Humanoid Robotics*, vol. 13, no. 01, 2016.

[22] C. Knabe, R. Griffin, J. Burton, G. Cantor-Cooke, L. Dantanarayana, G. Day, O. Ebeling-Koning, E. Hahn, M. Hopkins, J. Neal, *et al.*, "Designing for compliance: ESCHER, Team VALOR's compliant biped," *Journal of Field Robotics*, submitted.