IEEE International Conference on Robotics and Automation (ICRA), 2014

Autonomous Environment Manipulation to Assist Humanoid Locomotion

Martin Levihn Koichi Nishiwaki Satoshi Kagami Mike Stilman

***Abstract*— Legged robots have unique capabilities to traverse complex environments by stepping over and onto objects. Many footstep planners have been developed to take advantage of these capabilities. However, legged robots also have inherent constraints such as a maximum step height and distance. These constraints typically limit their reachable space, independent of footstep planning. Thus, we propose that robots such as humanoid robots that have manipulation capabilities should use them. A robot should autonomously modify its environment if necessary. We present a system that enabled a real robot to use a box to create itself a stair step or place a board on the ground to cross a gap, allowing it to reach its otherwise unreachable goal configuration.**

I. Introduction

Like humans, humanoid robots have the capability to step over and onto objects. However, also like humans, robots have inherent constraints such as a maximum step height and distance. Humans overcome their limitations by *modifying their environments*. Robots should do the same thing.

Consider grabbing a box to stand on to reach up to a bookshelf or placing a board over a gap in order to step over it. These kind of reasoning patterns become essential for robots that are expected to operate autonomously in complex, unstructured environments such as disaster areas. This work presents the first planning system to allow a humanoid robot to make such decisions autonomously. Figure 1 shows an example execution of a HRP-2 robot using the proposed system. The robot autonomously decides to place an object in front of the platform so that it can actually step up the platform to reach its goal configuration.

To achieve such behavior, we introduce the concept of *E*nvironmen*T* *A*ware *P*lanning (ETAP). ETAP lets the robot reason about *tapping* into resources available in the environment to complete its task. ETAP gives a robot the capability to use objects in its environment to assist its locomotion capabilities.

In order to handle the exponential search space complexity resulting from considering environment modifications [15], our proposed ETAP system extends existing work in footstep planning with the concept of constraint relaxation. The footstep planner is allowed to violate robot constraints such as maximum step width or height if necessary. While, the resulting path might not be directly executable by a locomotion controller, it provides a heuristic for determining how the robot should use its environment to accomplish the task. For example, in Figure 1 the initial footstep plan contains a constraint violation at the edge of the platform. The plan requires the robot to step higher than it physically can. The proposed system uses this information to guide the robot to grab the box and place it in front of the platform.

M. Levihn (*levihn@gatech.edu*) and M. Stilman (*mstilman@cc.gatech.edu*) are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA.
K. Nishiwaki (*k.nishiwaki@aist.go.jp*) and S. Kagami (*s.kagami@aist.go.jp*) are with the Digital Human Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan.

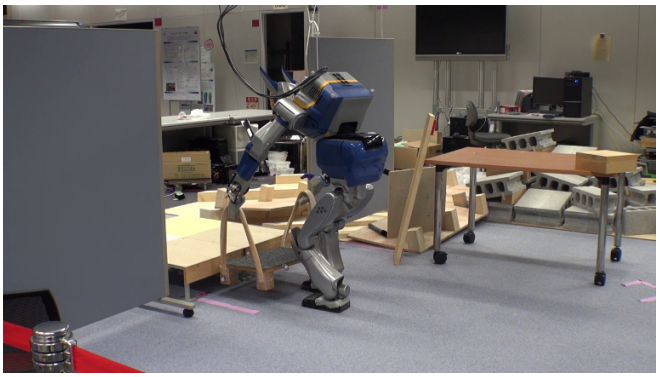


Fig. 1. Robot decides to place a box in front of the platform to be able to step up the platform.

The remainder of this work is organized as follows: Section II presents related work, and Section III provides the problem specification. The general overview of the proposed method in Section IV is made concrete with details about our actual implementation in Section V. After discussing the performed experiments in Section VI, the paper concludes with final remarks in Section VII.

II. Related Work

Over the last decades there have been significant advances in locomotion planning [1–7] and manipulation planning [8–11] for humanoid robots. However, to our knowledge manipulation has not been used to assist locomotion.

Given the complexity of humanoid robots, considering all degrees of freedom of the system during motion planning is typically limited to short-term motions [8]. Instead, for long horizon navigation tasks one common approach is to reduce the search space dimensionality by separating the footstep planning from the locomotion controller [1–4]. We adopt this approach in this work. Typically the footstep planner determines a sequence of steps to be executed by the robot and the locomotion controller then tracks these steps. Different search techniques have been explored for this general concept. While [1] utilizes A* search and an adaptive action set, [5] uses a sampling based search technique and half-steps. [2] utilizes D* Lite and continuous footstep

locations. In addition, the work presented in [2] introduces methods for efficient collision checking for footsteps. [7] builds on anytime repairing A* (ARA*) and randomized A* (R*) planners to reduce planning time while providing sub-optimality bounds. [6] introduces a bounding box method for footstep planning that sill allows the robot to step over objects. While all of these methods are complementary to our work, they treat the environment as something given that can not be modified by the robot. They do not take full advantage of the capabilities of a humanoid robot.

The concept of autonomous environment modification to assist a robot's locomotion was introduced by Wilfong [12]. The first practical planner was proposed by Chen [13]. Stilman and Okada presented this domain as Navigation Among Movable Obstacles (NAMO) for humanoid robots [14–17]. This domain enables a robot to move objects out of its way to clear a path to its goal configuration. NAMO planning systems were successfully executed on the HRP-2 robot [18, 19]. However, in all of these methods the robot preserves environment objects as hostile entities, that hinder its locomotion. In contrast, we propose that the robot should perceive environment objects as tools that could assist its locomotion.

III. Problem Specification

We specify the environment as the following set of entities:

- R - a humanoid robot,
- $\mathcal{S}$ - a set of static objects,
- $\mathcal{O}$ - a set of manipulable objects.

The robot's task is to reach a goal configuration $goal$ and it is allowed to manipulate the environment to its advantage.

This work assumes full world knowledge. This includes the robot's position as well as all object locations and properties (see below for details). Additionally, this work focuses on cases where a single object manipulation is sufficient to resolve a single constraint violation. However, the same object may be used to resolve multiple constraint violations during the entire execution.

IV. Algorithm

The algorithm takes advantage of existing research in humanoid locomotion planning and adopts the common approach of separating footstep planning from the locomotion controller [1]. However, to allow the robot to use environment objects as tools these methods have to be extended.

Considering all possible environment configurations is exponentially complex in the number of objects [15]. In order to resolve this complexity we apply the concept of constraint relaxed planning. The planner is allowed to violate the constraints of the robot in order to guide the decision making towards useful environment configurations. By creating a plan that violates the constraints, the planner can hypothesize a scenario where the robot would be able to follow a footstep plan to the goal, if only the environment was locally modified. This allows the planner to focus on these local modifications. To prevent the planner from unnecessarily violating robot constraints, a heuristic cost penalty is applied for each constraint violation. The magnitude of this penalty determines the planner's willingness to explore detours before considering environment modifications.

Algorithm 1: ETAP

```
   Input: R: robot, S: static objs, O: manipl. objs, g: goal
1  while R not at goal do
2      UPDATE(); // update robot and obj loc
3      moPl ← CONSTR_RELAXED_PLANNER(R, g);
4      if moPl contains constraint violations then
5          v ← GET_FIRST_VIOLATION(moPl);
           // get obj and obj target:
6          (o, o_t) ← RESOLVE_VIOLATION(v, R, O);
7          if o is NULL then
8              return; // goal not reachable
9          o_g ← GET_GRASP_POS(R, o);
10         pickPl ← LOCOMOTION_PLANNER(R, o_g);
11         R.execute(pickPl);
12         UPDATE();
13         grasp ← GET_GRASP_MOTION(R, o);
14         R.execute(grasp);
15         UPDATE();
16         dropPl ← LOCOMOTION_PLANNER(R, o_t);
17         R.execute(dropPl);
18         UPDATE();
19         drop ← GET_DROP_MOTION(R, o, o_t);
20         R.execute(drop);
21     else
22         R.execute(moPl);
```

Inevitable inaccuracies in actuation that occur on a real robot system make it difficult to exactly execute a detailed long horizon plan [20]. If the robot is planning detailed motions based on future environment configurations, the robot risks executing actions not supported by the environment. We therefore interleave planning and execution in our system [21]. Later modifications are not planned for until after the robot completed its current modification and the actual resulting environment configuration is determined[1].

Algorithm 1 summarizes the proposed ETAP framework. The algorithm begins with calling a constraint relaxed humanoid locomotion planner (line 3). In case the resulting plan violates constraints, an object and a target location to resolve the first violation are determined (line 4 - 6). The robot then moves to a grasp position, updates its configuration and grasps the object (line 9 - 14). Again updating the configurations the robot moves to the object target location and places the object (line 15 - 20). The robot repeats this procedure until it is at the goal.

[1]An alternative method is to trigger re-planning if too much divergence from the plan occurs. However we empirically found that this is too expensive in practice for the long horizon tasks considered in this work and requires substantial tuning of the re-planning thresholds.

V. IMPLEMENTATION

While the previous section provided a general overview of the proposed system, this section provides details of our implementation.

A. Constrained Relaxed Planner

The algorithm presented above requires a humanoid locomotion planner extended with the concept of constraint relaxation. We build on the planner presented in [1]. The planner performs an A* search through the space of possible footstep actions and locally adapts the action set if some actions are not possible due to environment properties. To allow the planner to violate robot constraints if necessary, we extend the action set the planner uses. We add actions for stepping substantially higher and further than the actual robot can do. To avoid unnecessary environment modifications, these actions are assigned a high cost. The increased cost of these actions leads the footstep planner to first explore longer environment paths before considering paths that include constraint violating actions.

B. Constraint Resolution

To resolve a constraint violation, an appropriate environment object and target location need to be chosen. In analogy to affordance [22], and similar to [2], we include a task-related object attribute representation. In our implementation, each object contains the following attributes:

- (x, y, z, θ): position and orientation,
- $weight$: the object's weight,
- $maxW$: the max weight the object can support,
- $\mathcal{FS}$: a set of flat surfaces the robot could potentially step on defined relative to the object center,
- $\mathcal{S}$: a set of surfaces defined relative to the object center that have to be supported by flat ground,
- $maxD$: the max height difference that is allowed between the surfaces in S,
- $\mathcal{GC}$: a set of possible grasp configurations for the object defined relative to the object center,
- $\mathcal{G}$: a set of grasp primitives for the object
- $\mathcal{D}$: a set of drop primitives for the object

For a given constraint violation, the algorithm first selects all manipulable objects that could potentially be used to modify the environment such that the robot could traverse the according location. In a pre-processing step the algorithm rejects all objects that do not fulfill the minimum requirements to support the robot. This includes support weight and the size of the biggest surfaces in $\mathcal{FS}$ as well as all objects that are too heavy for the robot to manipulate. The remaining objects are then further filtered according to the specific violation at hand. For example, if the robot is required to step higher than it can, only objects whose height is such that the robot can step on it and can step the remaining height are preserved. Similarly, if the violation required the robot to step further than supported, the algorithm only keeps objects whose largest side of any surface in $\mathcal{FS}$ is at least as large as the required step width. All remaining candidate objects are then sorted based on distance to the current robot configuration.

Algorithm 2: RESOLVE_VIOLATION

Input: v: violation, R: robot, O: manipl. objs
Output: o: suggested obj, o_t: suggested target loc

```
candidates ← {o ∈ O|o can be used to resolve constraint};
SORT_BY_DIST(candidates, R);
for o ∈ candidates do
    if R can not reach o then
        continue;// skip o
    o_t ← FIND_TARGET_LOCATION(o, v);
    if o_t is NULL then
        continue; // skip o
    return (o, o_t);
return (NULL, NULL); // Failure
```

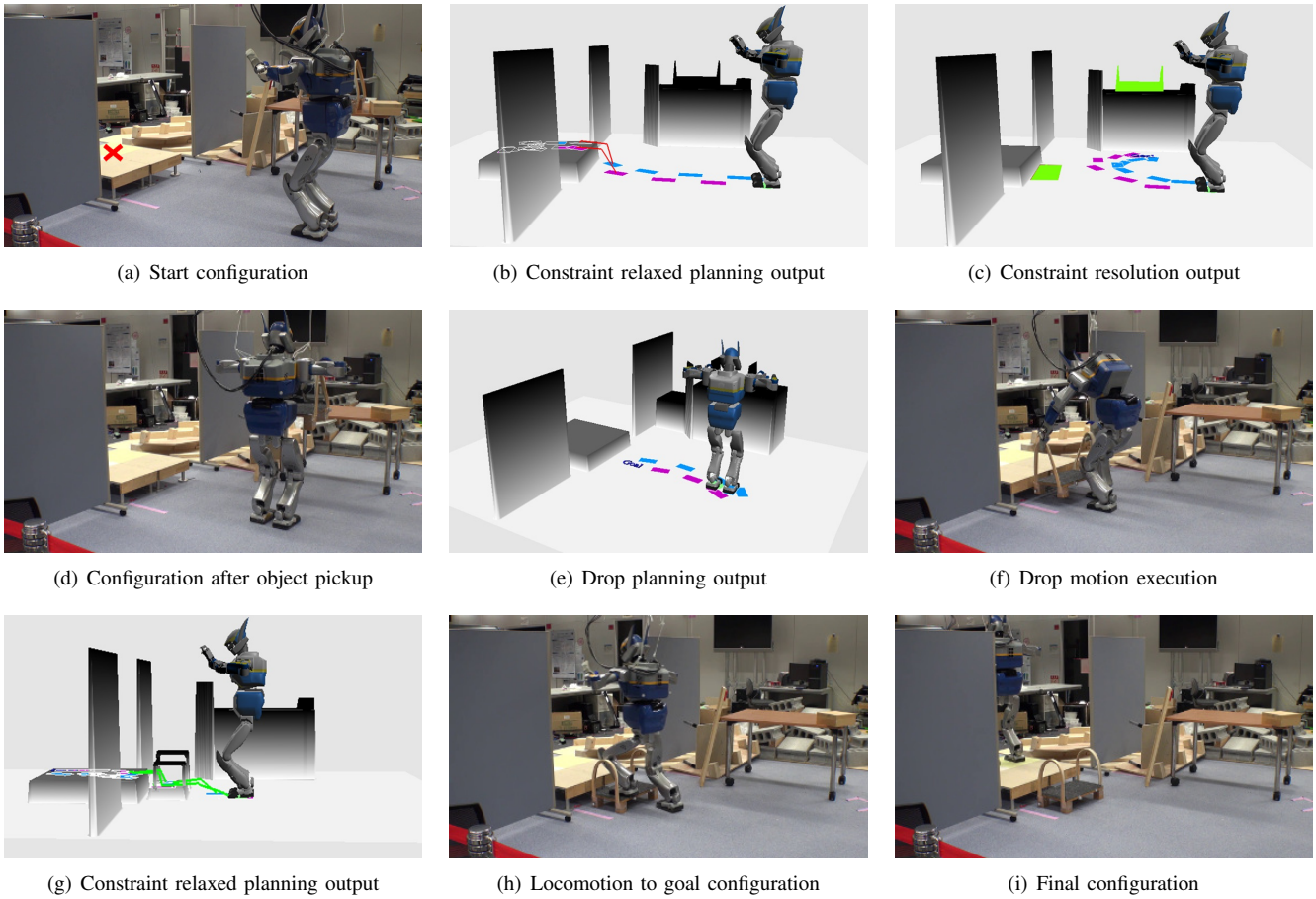
The sorted list is then iterated through and the first object that can successfully be reached by the robot and placed at the constraint violation location is returned. To verify reachability of an object the algorithm iterates through G and calls a non-modified version of the footstep planner presented in [1][2]. In order to find a target location for the object, the planner utilized the 2.5D grid map used by the footstep planner. The planner performs a local search around the coordinates of the violation and the last valid action within the actual plan and attempts to place the object. A successful object placement is achieved if all surfaces defined in S are supported by flat ground with a height difference of less then $maxD$. The local search perimeter is set such that a placement at any location within the perimeter would still resolve the constraint violation. We typically set the perimeter size equal to a robot step distance. For example, this would allow the planner to place the object at most step distance from the platform in Figure 1, still allowing the robot to step on the platform if on top of the box. If the object is either not reachable or no target location can be found, the next object in the list is checked[3]. This procedure is summarized in Algorithm 2.

C. Object Grasping and Dropping

If the robot has reached the grasp configuration for an object it needs to grasp the object. Similarly if it has reached the drop location for an object it needs to drop the object. To achieve this the planner simply selects a motion primitive according to its configuration relative to the object or target location.

[2]As we are only interested in verifying reachability at this point, we use a relatively large goal threshold here to reduce planning time.

[3]Our implementation does not verify reachability between the object grasp configuration and the drop location. We assume that the robot can at least return to its current location with the object and then move to the drop location. However, it is straight forward to extend the algorithm to also verify drop location reachability directly.



(a) Start configuration (b) Constraint relaxed planning output (c) Constraint resolution output

(d) Configuration after object pickup (e) Drop planning output (f) Drop motion execution

(g) Constraint relaxed planning output (h) Locomotion to goal configuration (i) Final configuration

Fig. 2. Proposed humanoid locomotion planning system: Stairs Example.

In our implementation the motion primitives are sets of precomputed motions defined as empirically sampled joint-space configurations. The joint angles are determined for different objects and grasp configurations. To execute a specific primitive, a spline-interpolated trajectory of the according joint angles is computed. The trajectories are tracked by PID control. The weight of the object is then incorporated or removed from the robot model respectively.

VI. Experiments

To verify the proposed algorithm, we implemented it on an actual robot system and in simulation.

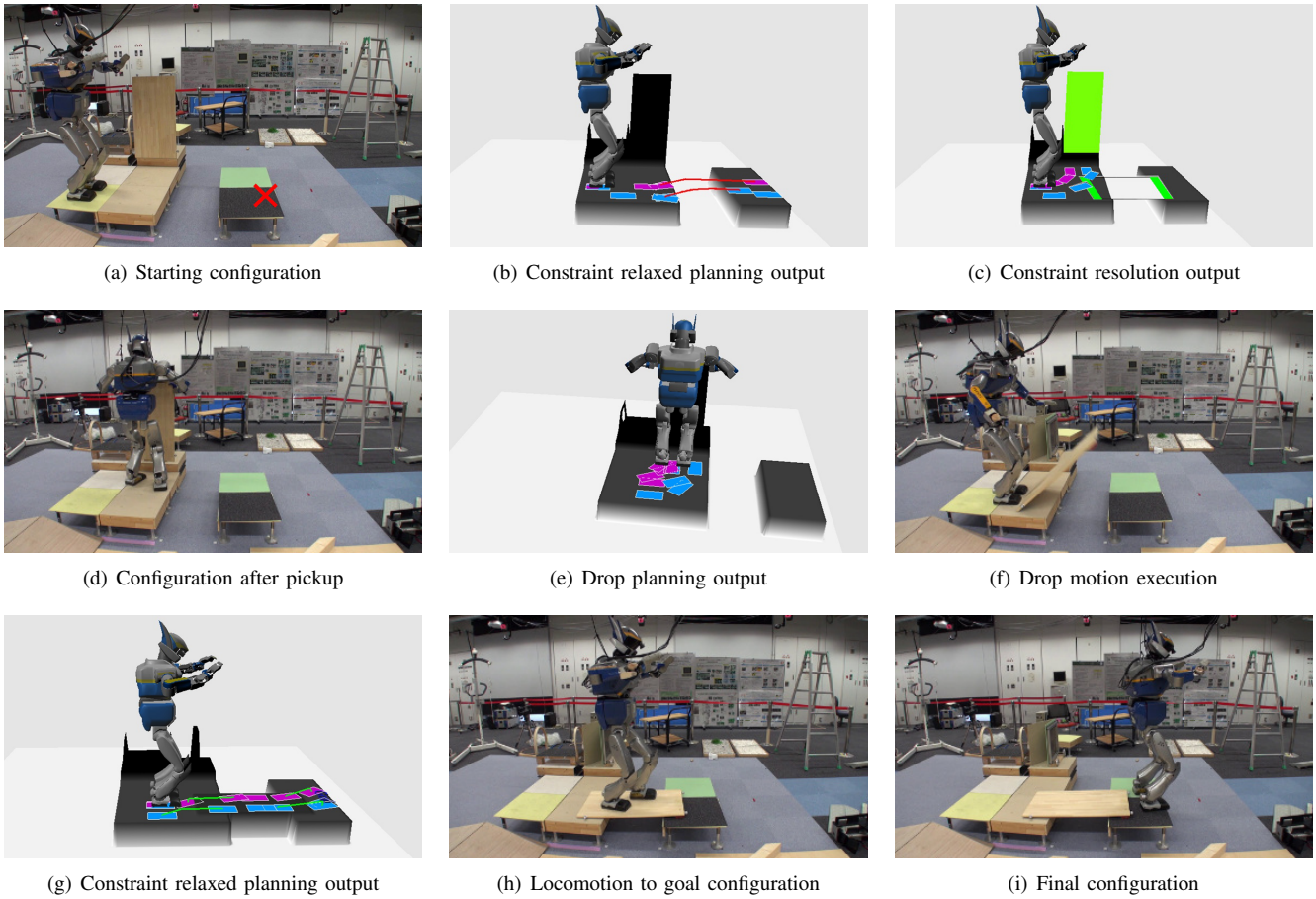
A. Real Robot

We implemented our algorithm on the HRP-2 robot platform and utilized existing locomotion and balance controllers for the robot [23] in our experiments. We also used a real-time motion capture system [24] to localize the robot and all objects within the $25m^2$ workspace.

We tested our system in different scenarios.

1) Stairs: The first scenario can be seen in Figure 2. A video of the execution can be accessed at `http://www.cc.gatech.edu/~mlevihn3/etap/`. In this setup the robot was commanded to take on a configuration at the end of the platform in front of it, as visualized in Figure 2(a). The platform was higher than the robot can step.

Figure 2(b) visualizes the output from the constraint relaxed planning step (Algorithm 1, line 3). The constraint relaxed planning step found a path to the goal, however it required the robot to step up higher than it physically can. The constraint resolution step (Algorithm 1, line 6) then determined an object to resolve the constrained violation at hand as well as a suitable target location, visualized in Figure 2(c). The planner picked the box as it could support the robot and had the correct height to allow the robot to step on the box itself and consecutively on the platform. The entire bottom surface of the box was required to be placed on flat ground. The robot was then guided to pickup the selected object (Algorithm 1, line 9 - 14). Once the robot picked up the object (Figure 2(d)) and updated its and the object's configurations, the robot was guided to the target location for the object and executed the appropriate drop motion primitive (Algorithm 1, line 15 - 20), visualized in Figure 2(e) - 2(f). The algorithm then looped and tried again to find a path to the goal given the new environment configuration. Figure 2(g) shows the output of the constraint relaxed planning step. This time a path to the goal was found without any constraint violations and the system



(a) Starting configuration (b) Constraint relaxed planning output (c) Constraint resolution output

(d) Configuration after pickup (e) Drop planning output (f) Drop motion execution

(g) Constraint relaxed planning output (h) Locomotion to goal configuration (i) Final configuration

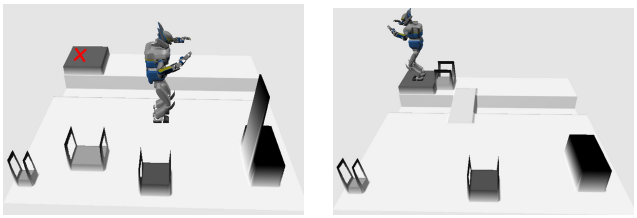
Fig. 3. Proposed humanoid locomotion planning system: Bridge Example.

guided the robot to the goal configuration without any further environment modifications (Figure 2(h) - 2(i)).

2) Bridge: Figure 3 as well as the accompanying video show a second execution example. In this setup the robot was tasked with reaching the other platform. The gap between the platforms was too wide for the robot to step over. The constraint relaxed planning steps determined that the robot needs to cross the gap in order to reach the goal. The planning system then guided the robot to pick up the board and drop it across both platforms. The planner picked the board as it was longer than the gap. Additionally, the board only required 8cm on each side to be placed on flat ground if dropped. This allowed the robot to drop it across both platforms. After the robot verified that it could reach the goal given the new environment configuration, the robot stepped over the board and reached its goal configuration.

B. Simulation

We additionally performed simulated experiments over 10 different domains with varying sizes and difficulties. The domains contained between 3 and 18 objects and the robot typically needed to perform between 1 and 3 environment modifications to reach its goal configuration. Figure 4 shows an example domain that required the robot to perform 2 environment modifications to reach its goal.



(a) Starting configuration (b) Final configuration

Fig. 4. Simulation visualisation. The robot is tasked with getting on top of the platform. The platform is higher than the robot can step. Additionally it can not step over gap. The robot placed the board over the gap and used a box to step up the platform.

The average computation time for the constraint relaxed planning step was 30.36s when no path without constraint violations existed and 3.06s otherwise (typically after the robot modified the environment). We can see that if no direct path to the goal existed this step took roughly 10x as long as if a direct path existed. This is because we used a very high cost for actions that violate robot constraints to avoid unnecessary environment modifications. Consequently the planner explored a larger space before using the constraint violating actions. If no constraint violations were necessary, the planner could explore the space more efficiently.

The average time for determining an object to resolve a constraint violation was 2.82s. The planning times to the grasp or drop location was 8.69s. Interestingly, the average planning times for a path to a pickup or drop location were almost 3x higher than for the constraint relaxed planning case if no constraint violations were necessary despite the planner behaving similar in these cases. This was caused by the fact that our implementation required a substantially higher goal configuration accuracy for pickup or drop location planning than for the final goal configuration.

The average execution time for the robot including pickup and drop motions was 84.52s, dominating the planning time.

VII. Conclusion

We presented the novel domain of Environment Aware Planning. This new domain gives a robot the option to manipulate its environment in an effort to assist its locomotion. We presented a planning system for it that was successfully executed on a real robot. The proposed system enabled the robot to reach a goal configuration outside of its reachable space by building itself a stair step and a bridge. To the best of our knowledge this is the first time a humanoid robot showed such a behavior fully autonomously.

Future work will focus on techniques to allow the use of multiple objects to resolve a single constraint violation. We also plan to extend the proposed system's applicability to a variety of related scenarios. For example, the same methods described here can be used to allow the robot to decide that specific surface properties need to be modified. The robot could decide to place a board over uneven surface to allow itself to traverse the area more safely.

The kind of reasoning methods described in this work are vital for a robot to operate autonomously in unstructured environments such as disaster areas.

Acknowledgements

This work was supported in part by the ONR under Grant N000141210143. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research. We also gratefully acknowledge support from the Japan Society for the Promotion of Science.

References

[1] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami, "An adaptive action model for legged navigation planning," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*. IEEE, 2007, pp. 196–202.

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

[3] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," in *IEEE Int. Conf. Hum. Rob., Munich, Germany*, 2003.

[4] Y. Ayaz, A. Konno, K. Munawar, T. Tsujita, and M. Uchiyama, "Planning footsteps in obstacle cluttered environments," in *Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on*. IEEE, 2009, pp. 156–161.

[5] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *Robotics, IEEE Transactions on*, vol. 28, no. 2, pp. 427–439, 2012.

[6] N. Perrin, O. Stasse, F. Lamiraux, Y. J. Kim, and D. Manocha, "Real-time footstep planning for humanoid robots among 3d obstacles using a hybrid bounding box," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 977–982.

[7] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012.

[8] J. J. Kuffner Jr, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Autonomous Robots*, vol. 12, no. 1, pp. 105–118, 2002.

[9] M. Gienger, M. Toussaint, and C. Goerick, "Task maps in humanoid robot manipulation," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 2758–2764.

[10] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [grand challenges of robotics]," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, pp. 20–29, 2007.

[11] K. Hauser, V. Ng-Thow-Hing, and H. Gonzalez-Baños, "Multi-modal motion planning for a humanoid robot manipulation task," in *Robotics Research*. Springer, 2011, pp. 307–317.

[12] G. Wilfong, "Motion planning in the presence of movable obstacles," in *SCG '88: Proceedings of the fourth annual symposium on Computational geometry*. New York, NY, USA: ACM, 1988, pp. 279–288.

[13] P. C. Chen and Y. K. Hwang, "Practical path planning among movable obstacles," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 444–449.

[14] K. Okada, A. Haneda, H. Nakai, M. Inaba, and H. Inoue, "Environment manipulation planner for humanoid robots using task graph that generates action sequence," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2004, pp. 1174–1179.

[15] M. Stilman and J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," in *IEEE/RAS International Conference on Humanoid Robotics*, November 2004, pp. 322–341.

[16] ——, "Planning among movable obstacles with artificial constraints," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1295–1307, 2008.

[17] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3327–3332.

[18] M. Stilman, K. Nishiwaki, S. Kagami, and J. J. Kuffner, "Planning and executing navigation among movable obstacles," *Advanced Robotics*, vol. 21, no. 14, pp. 1617–1634, 2007.

[19] Y. Kakiuchi, R. Ueda, K. Kobayashi, K. Okada, and M. Inaba, "Working with movable obstacles using on-line environment perception reconstruction using active sensing and color range sensor," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1696–1701.

[20] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[21] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1470–1477.

[22] J. Gibson, "The concept of affordances," *Perceiving, acting, and knowing*, pp. 67–82, 1977.

[23] K. Nishiwaki, J. Chestnutt, and S. Kagami, "Autonomous navigation of a humanoid robot on unknown rough terrain," in *Intl Symposium on Robotics Research*, 2011.

[24] M. Stilman, P. Michel, J. Chestnutt, K. Nishiwaki, S. Kagami, and J. Kuffner, "Augmented reality for robot development and experimentation," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-55, 2005.