Footstep Planning on Uneven Terrain with Mixed-Integer Convex Optimization

Robin Deits[1] and Russ Tedrake[2]

***Abstract*— We present a new method for planning footstep placements for a robot walking on uneven terrain with obstacles, using a mixed-integer quadratically-constrained quadratic program (MIQCQP). Our approach is unique in that it handles obstacle avoidance, kinematic reachability, and rotation of footstep placements, which typically have required non-convex constraints, in a single mixed-integer optimization that can be efficiently solved to its global optimum. Reachability is enforced through a convex inner approximation of the reachable space for the robot's feet. Rotation of the footsteps is handled by a piecewise linear approximation of sine and cosine, designed to ensure that the approximation never overestimates the robot's reachability. Obstacle avoidance is ensured by decomposing the environment into convex regions of obstacle-free configuration space and assigning each footstep to one such safe region. We demonstrate this technique in simple 2D and 3D environments and with real environments sensed by a humanoid robot. We also discuss computational performance of the algorithm, which is currently capable of planning short sequences of a few steps in under one second or longer sequences of 10-30 footsteps in tens of seconds to minutes on common laptop computer hardware. Our implementation is available within the Drake MATLAB toolbox [1].**

I. INTRODUCTION

The purpose of a footstep planner is to find a list of footstep locations that a walking robot can follow safely to reach some goal. Footstep planning is a significant simplification of motion planning through contact, one in which the whole-body kinematics and dynamics are typically coarsely approximated or ignored in order to produce a tractable problem. The challenge of footstep planning thus consists of finding a path through a constrained environment to a goal while respecting constraints on the locations of and displacements between footsteps. An example of such a plan is shown in Fig. 1.

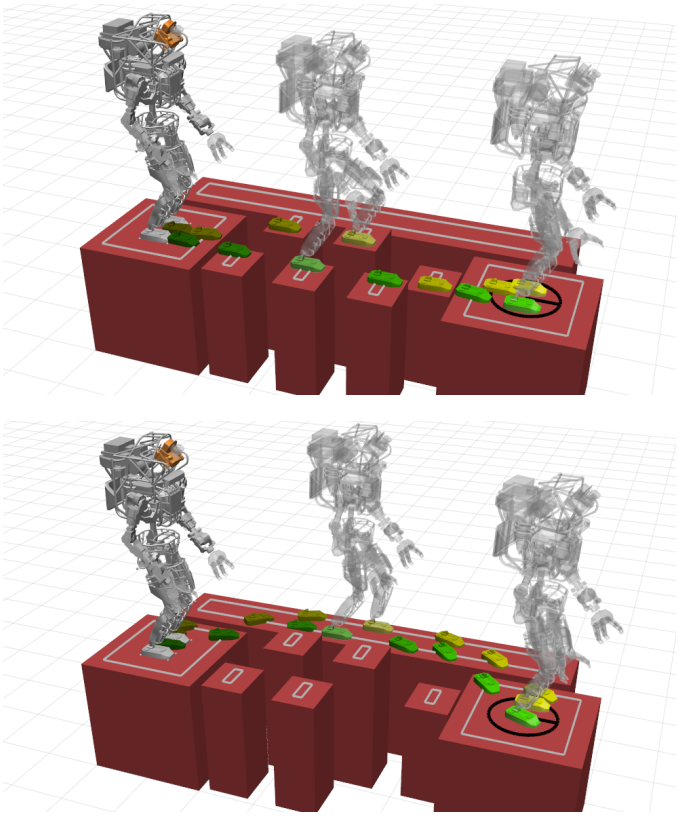


Fig. 1. Two examples of the output of our MIQCQP footstep planner. Above: An Atlas biped planning footsteps across a set of stepping stones. Below: With one stepping stone removed, Atlas must take the longer detour around the stepping stones. The gray rectangles are the boundaries of convex regions of obstacle-free configuration space generated by IRIS [2] projected into the xy plane.

Broadly speaking, there exist two families of approaches to footstep planning: discrete searches and continuous optimizations. The discrete search approaches have typically involved a precomputed action set: either represented as a set of possible displacements from one footstep to the next or a set of possible footholds in the environment. Chaining actions together forms a tree of possible footstep plans, which can be explored using existing discrete search methods like A^* and RRT. Action set approaches using pre-computed step displacements have been used by Michel [3], Baudouin [4], Chestnutt [5], [6], and Kuffner [7], [8]. Similarly, Shkolnik et al. used a precomputed set of dynamic motions, rather than foot displacements, and an RRT search to find motions for a quadruped [9]. The fixed foothold sets have been used by Bretl for climbing robots [10] and by Neuhaus for the LittleDog quadruped [11]. These approaches can easily handle obstacle avoidance by pruning the tree of actions when a particular action would put a foot in collision with an obstacle [7], [8], [4], including obstacle avoidance in the cost function evaluated at each leaf of the tree [5], [3], or adapting the set of actions when a collision is detected [6]. However, they have also tended to suffer from the tradeoff between a small action set, which reduces the branching factor of the search tree, and a large action set, which covers a larger set

This work was supported by the Fannie and John Hertz Foundation, the MIT Energy Initiative, MIT CSAIL, and the DARPA Robotics Challenge.

[1]Robin Deits is with the Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA rdeits@csail.mit.edu

[2]Russ Tedrake is with the Faculty of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, USA russt@csail.mit.edu

of the true space of foot displacements but is much harder to search [4]. In addition, applying A^* or other informed search methods to our problem is complicated by the difficulty in defining a good heuristic for partial footstep plans: we cannot generally know how many additional footsteps a partial plan will need in order to reach the goal without actually searching for those steps [5].

The continuous optimization approaches, on the other hand, operate directly on the poses of the footsteps as continuous decision variables. This avoids the restriction to a small set of fixed actions and thus allows more possible footstep plans to be explored, but correctly handling rotation and obstacle avoidance turns out to be difficult in a continuous optimization. Both footstep rotation and obstacle avoidance generally require non-convex constraints to enforce them, since the set of rotation matrices and the set of points outside a closed obstacle are non-convex. We typically cannot find guarantees of completeness or global optimality for such non-convex problems [12]. We have presented a non-convex continuous optimization for footstep planning, used by Team MIT during the DARPA Robotics Challenge 2013 Trials [13], but this optimization could not guarantee optimality of its solutions or find paths around obstacles. Alternatively, footstep rotation and obstacle avoidance can simply be ignored: Herdt et al. fix the footstep orientations and do not consider obstacle avoidance. This allows them to form a single quadratic program (QP) which can choose optimal footstep placements and control actions for a walking robot model [14].

We choose to use a mixed-integer convex program (specifically, a mixed-integer quadratically constrained quadratic program) to provide a more capable continuous footstep planner. Such a program allows us to perform a continuous optimization of the footstep placements, while using integer variables to absorb any non-convex constraints. We handle orientation of the footstep placements by approximating the trigonometric $\sin$ and $\cos$ functions with piecewise linear functions, using a set of integer variables to choose the appropriate approximation. We also avoid the non-convex constraints inherent in avoiding obstacles by instead enumerating a set of convex obstacle-free configuration space regions and using additional integer variables to assign footsteps to those regions. The presence of integer constraints significantly complicates our formulation, but a wide variety of commercial and free tools for mixed-integer convex programming exist, all of which can provide globally optimal solutions or proofs of infeasibility, as appropriate [15], [16], [17], [18]. Thus, we can solve an entire footstep planning problem to optimality while ensuring obstacle avoidance, a task that, to our knowledge, has not been accomplished before.

This is not unlike the work of Richards et al., who constructed a mixed-integer linear program to plan UAV trajectories while avoiding obstacles [19]. They represented each (convex, 2D) obstacle as a set of linear constraints, each of which generates a pair of half-spaces, one containing the obstacle and one not. They assigned a binary integer variable

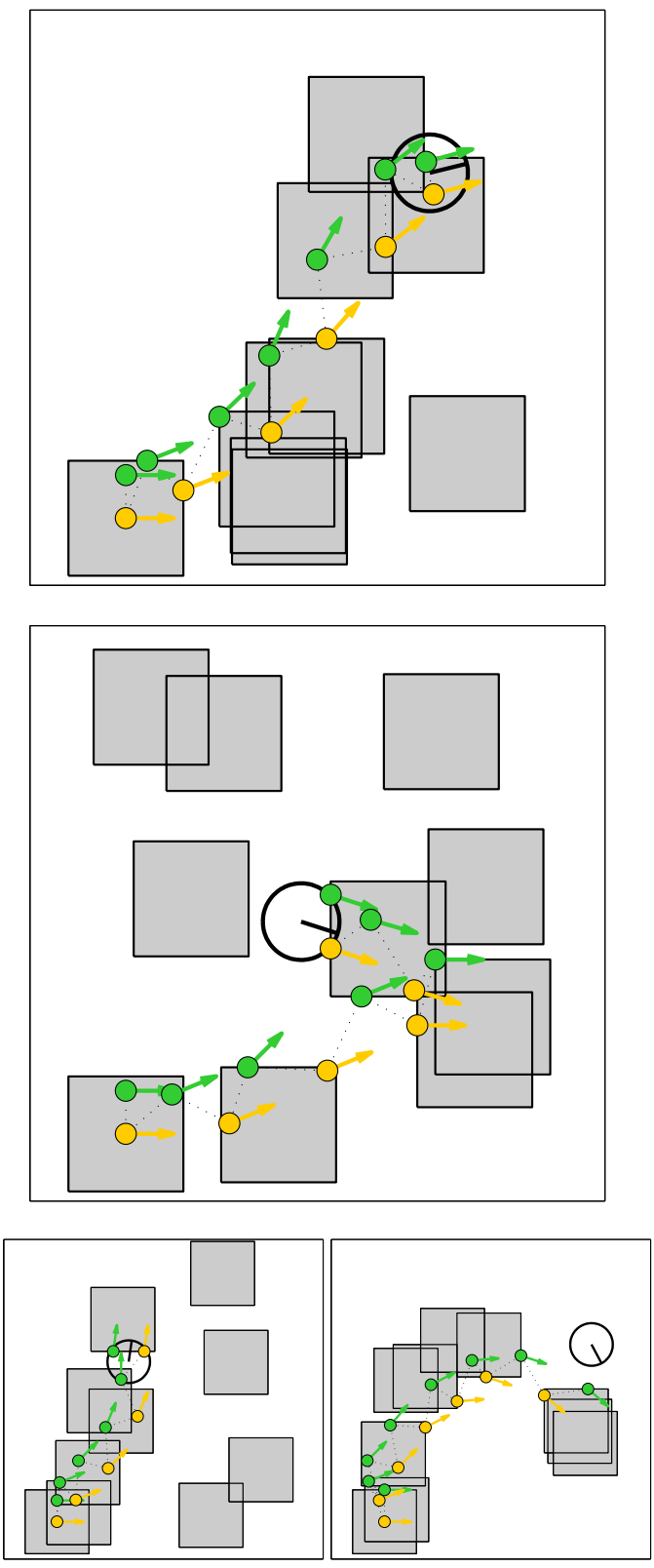


Fig. 2. Four randomly generated 2D environments demonstrating the MIQCQP footstep planner. The gray squares are the randomly placed regions of safe terrain; the black circle is the goal location, with a line indicating the desired orientation of the robot; and the green and yellow markers are the locations of the right and left footsteps, respectively, with arrows showing the orientation of each step. The foot is assumed to be a point for these examples.

to every pair of half-spaces and required that, for every obstacle, the vehicle's location must be in at least one of the non-obstacle half-spaces. Instead of adding binary variables for every single face of every obstacle, we precompute several convex obstacle-free regions, then assign a single binary variable to each region and require that every footstep is assigned to one such region, which dramatically reduces the number of binary variables required. Our ability to generate these convex obstacle-free regions relies on our recent development of IRIS, an algorithm for computing large convex regions of obstacle-free space [2].

II. TECHNICAL APPROACH

Our task is to determine the precise x, y, z and θ (yaw) positions of N footsteps, subject to the constraints that

1) Each step does not intersect any obstacle
2) Each step is within some convex reachable region relative to the position of the prior step

To accomplish this, we invert the non-convex problem of *avoiding* every obstacle and instead reformulate the problem as one of assigning each footstep to some pre-computed convex region of obstacle-free terrain. We then add quadratic constraints to ensure that each footstep is reachable from the prior step, using a piecewise linear approximation of $\sin$ and $\cos$ to handle step rotation.

A. Assigning Steps to Obstacle-Free Regions

Our first task is to decompose the 3D environment into a set of convex regions in which the foot can safely be placed. The IRIS algorithm, first presented in [2] and designed for this particular task, quickly generates obstacle-free convex regions in the x, y, and θ (yaw) configuration space of the robot's feet. Each of these obstacle free terrain regions is represented as a set of linear constraints defining a polytope in x, y, θ. Additionally, for each region we fit a plane in x, y, and z to the local terrain and add additional linear constraints to force footsteps in that region to lie on the plane. We label the total number of convex regions as R and for each region r we identify the associated linear constraints with the matrix A_r and vector b_r such that if footstep f_j is assigned to region r, then

$$A_r f_j \leq b_r$$

where

$$f_j \equiv \begin{bmatrix} x_j \\ y_j \\ z_j \\ \theta_j \end{bmatrix} \text{ and } r \in \{1, \ldots, R\}$$

To describe the assignment of footsteps to safe regions, we construct a matrix of binary variables $H \in \{0,1\}^{R \times N}$, such that if $H_{r,j} = 1$ then footstep j is assigned to region r:

$$H_{r,j} \implies A_r f_j \leq b_r \tag{1}$$

$$\sum_{r=1}^{R} H_{r,j} = 1 \quad \forall j = 1, \ldots, N \tag{2}$$

The *implies* operator in (1) can be converted to a linear constraint using a standard big-M formulation [20] or handled directly by mixed-integer programming solvers such as IBM ILOG CPLEX [17]. The constraint (2) requires that every footstep be assigned to exactly one safe terrain region.

B. Ensuring Reachability

We choose to approximate the reachable set of footstep positions as an intersection of circular regions in the xy plane, with additional linear constraints on footstep displacements in yaw and z. The reachable set defined by the intersection of two circular regions is shown in Fig. 3b. Other approaches have typically used polytope representations of the reachable set [13], [14], but such an approach results in a non-convex constraint when rotation is allowed, even under our piecewise linear approximation of $\sin$ and $\cos$.

Setting aside the question of footstep rotation for the moment, we can describe our reachable region with a set of convex quadratic constraints. For each footstep j, we require that

$$\left\| \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \left(\begin{bmatrix} x_{j-1} \\ y_{j-1} \end{bmatrix} + \begin{bmatrix} \cos\theta_{j-1} & -\sin\theta_{j-1} \\ \sin\theta_{j-1} & \cos\theta_{j-1} \end{bmatrix} p_1 \right) \right\| <= d_1 \tag{3}$$

$$\left\| \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \left(\begin{bmatrix} x_{j-1} \\ y_{j-1} \end{bmatrix} + \begin{bmatrix} \cos\theta_{j-1} & -\sin\theta_{j-1} \\ \sin\theta_{j-1} & \cos\theta_{j-1} \end{bmatrix} p_2 \right) \right\| <= d_2 \tag{4}$$

where p_1, p_2 are the centers of the circles, expressed in the frame of footstep $j-1$ and d_1, d_2 are their radii. When θ_{j-1} is fixed, constraints (3) and (4) are convex quadratic constraints, but including θ as a decision variable makes the constraint non-convex. We will handle this problem by introducing two new variables for every footstep: s_j and c_j, which will approximate $\sin\theta_j$ and $\cos\theta_j$, respectively. Constraints (3,4) thus become:

$$\left\| \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \left(\begin{bmatrix} x_{j-1} \\ y_{j-1} \end{bmatrix} + \begin{bmatrix} c_j & -s_j \\ s_j & c_j \end{bmatrix} p_1 \right) \right\| <= d_1 \tag{5}$$

$$\left\| \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \left(\begin{bmatrix} x_{j-1} \\ y_{j-1} \end{bmatrix} + \begin{bmatrix} c_j & -s_j \\ s_j & c_j \end{bmatrix} p_2 \right) \right\| <= d_2. \tag{6}$$

Since p_1, p_2, d_1, d_2 are fixed, this is still a convex quadratic constraint.

Our work is not yet complete, however, since we now must enforce that s_j and c_j approximate $\sin$ and $\cos$ without introducing non-convex trigonometric constraints. We choose instead to create a simple piecewise linear approximation of $\sin$ and $\cos$ and a set of binary variables to determine which piece of the approximation to use. We construct binary matrix $S \in \{0,1\}^{L \times N}$, where L is the number of piecewise linear segments and add constraints of the form

$$S_{\ell,j} \implies \begin{cases} \phi_\ell \leq \theta_j \leq \phi_{\ell+1} \\ s_j = g_\ell \theta_j + h_\ell \end{cases} \tag{7}$$

$$\sum_{\ell=1}^{L} S_{\ell,j} = 1 \quad \forall j = 1, \ldots, N \tag{8}$$

where g_ℓ and h_ℓ are the slope and intercept of the linear approximation of $\sin\theta$ between ϕ_ℓ and $\phi_{\ell+1}$. We likewise add piecewise linear constraints of the same form for c_j.

The particular choice of g_ℓ and h_ℓ turns out to be quite important: we must ensure that our approximation never overestimates the reachable space of foot placements. We can verify this empirically for the approximation shown in Fig. 3a by checking that the intersection of constraints (5,6) is contained within the intersection of constraints (3,4) for all values of θ. The reachable sets for footsteps at a variety of orientations are shown in Fig. 3c.

C. Determining the Total Number of Footsteps

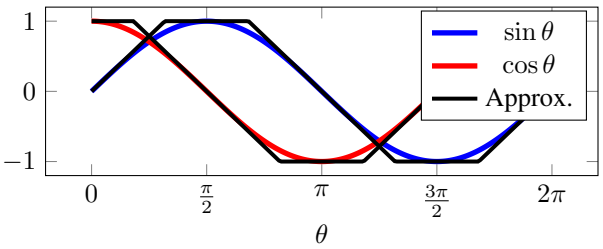
In general we cannot expect to know *a priori* the total number of footsteps which must be taken to bring the robot to a goal pose, so we need some method for determining N efficiently. We could certainly just repeat the optimization for different values of N, performing a binary search to find the minimum acceptable number of steps, but for efficiency's sake we would prefer to avoid the many runs of the optimizer that this would require.

We might attempt to determine the number of required steps by setting N sufficiently large and simply adding a cost on the squared distance from each footstep to the goal pose, which will stretch our footstep plan towards the goal. If the footsteps reach the goal before N steps have been taken, then we can trim off any additional steps at the end of the plan. However, this approach allows the footstep planner to produce strides of the maximum allowable length for every footstep, even on obstacle-free flat terrain. During experiments leading up to the DARPA Robotics Challenge trials, we determined that a forward stride of 40 cm was achievable on the Atlas biped, but that a nominal stride of approximately 20 cm was safer and more stable. We would thus like to express in our optimization a preference for a particular nominal stride length while still allowing occasional longer strides needed to cross gaps or clear obstacles. We can try to create this result by adding additional quadratic costs on the relative displacement between footsteps, but this requires very careful tuning of the weights of the distance-to-goal cost and the relative step cost for each individual step in order to ensure that the costs balance precisely at the nominal step length for each step.

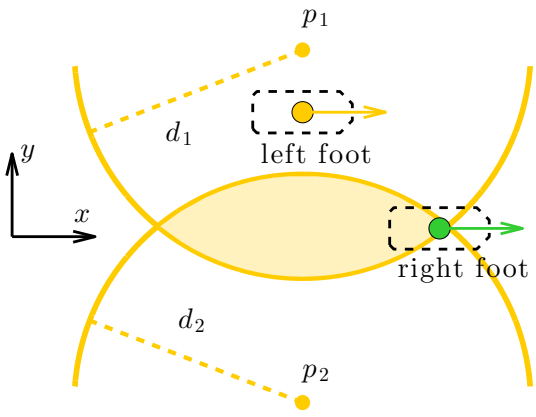
Instead, we choose a much simpler cost function, with a quadratic cost on the distance from the last footstep to the goal and identical cost weights on the displacement from each footstep to the next. To control the number of footsteps used in the plan, and thus the length of each stride, we add a single binary variable to each footstep, which we will label as t_j (for 'trim'). If t_j is true, then we require that step j be fixed to the initial position of that same foot:

$$t_j \implies f_j = \begin{cases} f_1 & \text{if } j \text{ is odd} \\ f_2 & \text{if } j \text{ is even.} \end{cases} \tag{9}$$

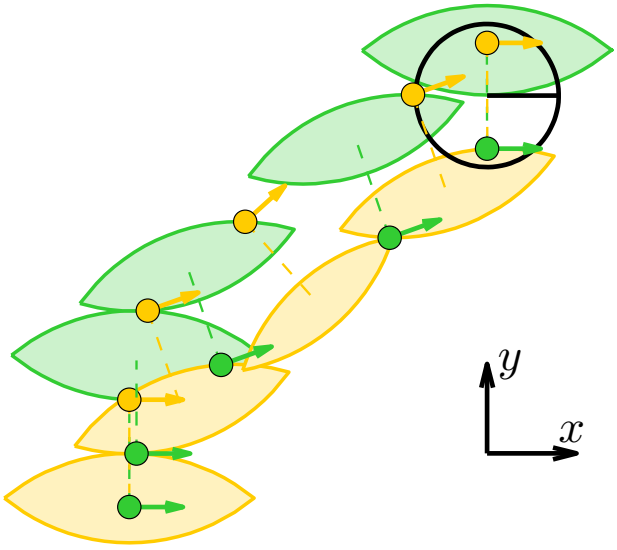
Note that f_1 and f_2 are the *fixed* current positions of the robot's two feet.



(a) Piecewise linear approximation of sine and cosine



(b) Approximation of the reachable set of locations for the right foot, given the position of the left foot. The gold arrow shows the position and orientation of the left foot, viewed from above. The shaded region shows the set of reachable poses for the right foot in the xy plane, defined as the intersection of constraints (5,6) at orientation of $\theta_j = 0$, for which our approximation of sin and cos is exact. One possible future pose of the right foot is shown for reference.



(c) A simple footstep plan with 2D reachable regions shown. The goal is the black circle in the top right, and each arrow shows the position and orientation of one footstep. For each step, we draw the shaded region defined by constraints (5,6) into which the next step must be placed. Note that the feasible region shrinks when the step orientation is not a multiple of $\pi/2$, as our approximation of sin and cos becomes inexact.

Since each footstep for which $t_j = 1$ is fixed to the current position of the robot's feet, the number of footsteps which are actually used to move the robot to the goal is $N - \sum_{j=1}^{N} t_j$. By assigning a negative cost value to each binary variable t_j, we can create an incentive to reduce the number of footsteps used in the plan. To tune the nominal stride length, we can simply adjust the cost assigned to the t_j. Increasing the magnitude of this cost will lengthen the nominal stride uniformly, and decreasing the magnitude will shorten the stride. Thus, we have a single value to tune in order to set the desired stride length, while still allowing strides which exceed this length. After the optimization is complete, we can remove any footsteps at the beginning of the plan for which t_j is true.

D. Complete Formulation

Putting all of the pieces together gives us the entire footstep planning problem:

$$\underset{f_1,\ldots,f_j,S,C,H,t_1,\ldots,t_j}{\text{minimize}} (f_N - g)^\top Q_g (f_N - g) + \sum_{j=1}^{N} q_t t_j + \sum_{j=1}^{N-1} (f_{j+1} - f_j)^\top Q_r (f_{j+1} - f_j))$$

subject to, for $j = 1, \ldots, N$

safe terrain regions:

$$H_{r,j} \implies A_r f_j \leq b_r \qquad r = 1, \ldots, R$$

piecewise linear $\sin\theta$:

$$S_{\ell,j} \implies \begin{cases} \phi_\ell \leq \theta_j \leq \phi_{\ell+1} \\ s_j = g_\ell \theta_j + h_\ell \end{cases} \qquad \ell = 1, \ldots, L$$

piecewise linear $\cos\theta$:

$$C_{\ell,j} \implies \begin{cases} \phi_\ell \leq \theta_j \leq \phi_{\ell+1} \\ c_j = g_\ell \theta_j + h_\ell \end{cases} \qquad \ell = 1, \ldots, L$$

approximate reachability:

$$\left\| \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \left(\begin{bmatrix} x_{j-1} \\ y_{j-1} \end{bmatrix} + \begin{bmatrix} c_j & -s_j \\ s_j & c_j \end{bmatrix} p_i \right) \right\| <= d_i \quad i = 1, 2$$

fix extra steps to initial pose:

$$t_j \implies f_j = \begin{cases} f_1 & \text{if } j \text{ is odd} \\ f_2 & \text{if } j \text{ is even.} \end{cases}$$

$$\sum_{r=1}^{R} H_{r,j} = \sum_{\ell=1}^{L} S_{\ell,j} = \sum_{\ell=1}^{L} C_{\ell,j} = 1$$

$$H_{r,j}, S_{\ell,j}, C_{\ell,j}, t_j \in \{0, 1\}$$

bounds on step positions and differences:

$$f_{\text{min}} \leq f_j \leq f_{\text{max}}$$

$$\Delta f_{\text{min}} \leq (f_j - f_{j-1}) \leq \Delta f_{\text{max}}$$

where $g \in \mathbb{R}^4$ is the x, y, z, θ goal pose, $Q_g \in \mathbb{S}^4_+$ and $Q_r \in \mathbb{S}^4_+$ are objective weights on the distance to the goal and between steps, $q_t \in \mathbb{R}$ is an objective weight on trimming unused steps, and $f_{\text{min}}, f_{\text{max}}, \Delta f_{\text{min}}, \Delta f_{\text{max}} \in \mathbb{R}^4$ are bounds on the absolute footstep positions and their differences, respectively. We also fix f_1 and f_2 to the initial poses of the robot's feet.

E. Solving the Problem

We have implemented this approach in MATLAB [21], using the commercial solver Gurobi [15] to solve the MIQCQP itself. Typical problems involving 10 to 20 footsteps and 10 convex safe terrain regions can be solved in a few seconds to one minute on a Lenovo laptop with an Intel i7 clocked at 2.9 GHz. Smaller footstep plans involving just a few steps can be solved in well under one second on the same hardware, so this method is capable of providing short-horizon footstep plans at realtime rates while walking or longer footstep plans involving complex path planning while stationary. The Mosek and CPLEX optimizers [16], [17] were also capable of solving the problem, but we generally found that Gurobi found optimal solutions or proofs of infeasibility more quickly in our experiments.

III. RESULTS

To demonstrate the MIQCQP footstep planning algorithm, we first generated a collection of random 2D environments. Each environment consisted of 10 square regions of safe terrain, 9 of which were uniformly randomly placed within the bounds of the environment, and 1 of which was placed directly under the starting location of the robot to represent the robot's currently occupied terrain. A goal pose was uniformly randomly placed within the xy bounds of the environment with a desired orientation between $\pm\frac{\pi}{2}$ relative to the robot's starting orientation. Several such environments and the resulting footstep plans are shown in Fig. 2. All the footstep plans shown in this paper are the result of convergence to within 0.1% or less of the globally optimal cost value, as reported by Gurobi.

Next, we manually generated several 3D example environments using Drake, a software toolbox for planning and control [1]. These environments and the resulting footstep plans are shown in Figs. 1, 3, 5. The IRIS algorithm [2] was used to generate convex regions in the configuration space of a very simple box model of the entire robot, shown in Fig. 4, in order to avoid collisions between the upper body and the environment. This approach is sufficient to generate rich behaviors such as turning sideways to move through a narrow gap, as in Fig. 5. Currently, we only ensure that each footstep admits a collision-free posture of the robot, but we do not account for collisions during the transitions between those postures; we will discuss possible ways to address this in Sect. IV.

We have also demonstrated the MIQCQP planner on real terrain, using sensor data collected by Atlas. To generate this plan, we captured LIDAR scans of a stack of cinderblocks like those encountered in the DRC Trials and constructed a heightmap of the scene using the perception tools developed by Team MIT for the DRC [13]. A Sobel filter was used to classify areas of the terrain which were steeper than a predefined threshold [22], and these steep areas were represented as obstacles for the footstep planner. We used

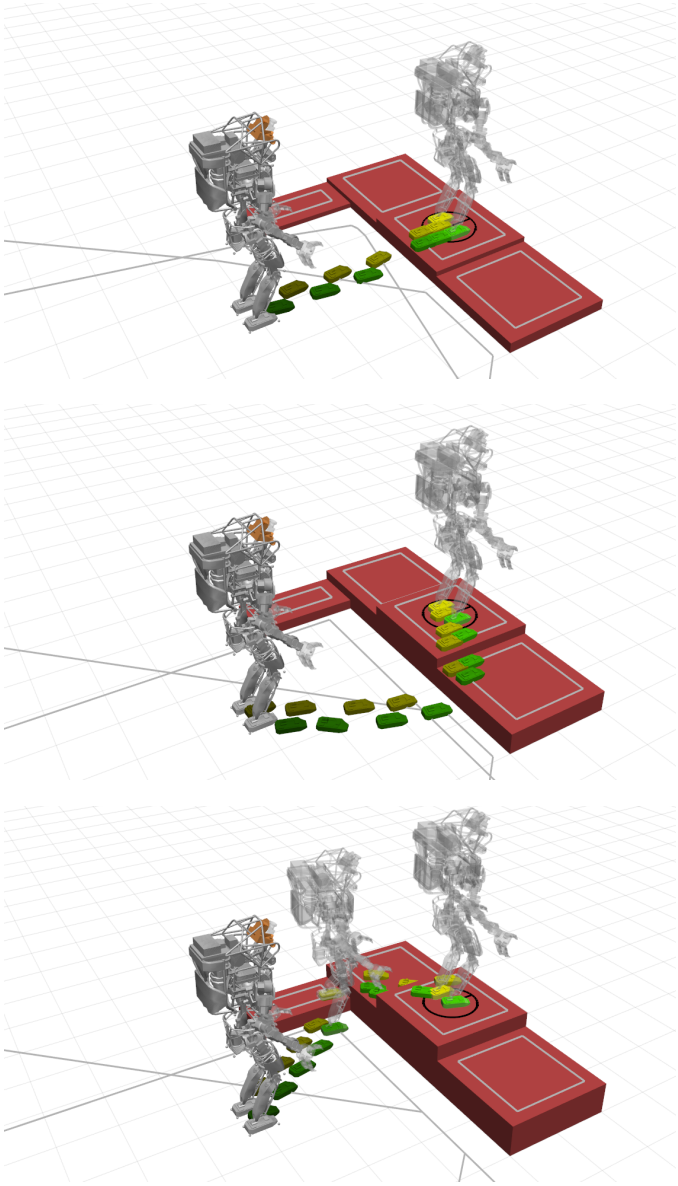


Fig. 3. Three similar environments, with footstep plans for each. Top: Atlas can mount the center pedestal in one stride. Middle: Raising the center pedestal to twice Atlas' maximum vertical stride forces the robot to detour to the right. Bottom: Raising the pedestals even further requires Atlas to use three platforms to get to the required height. Gray lines are the boundaries of the convex regions of obstacle-free configuration space, generated by IRIS.

the IRIS algorithm [2] to generate seven convex safe terrain regions which covered the terrain of interest in front of the robot. Several footstep plans to different goal poses using these regions are shown in Fig. 6.

IV. CONCLUSION AND FUTURE WORK

We have demonstrated a novel footstep planning approach, which replaces nonlinear, non-convex constraints with mixed-integer convex constraints. This allows us to solve footstep planning problems to their global optimum, within our linear approximation of rotation. We have inverted the problem of avoiding obstacles into a problem of assigning

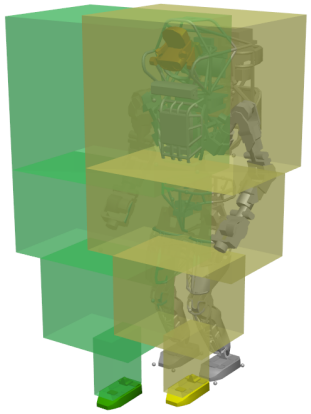


Fig. 4. Simplified bounding box model of the robot used to plan footstep locations that will be collision-free for the entire robot.

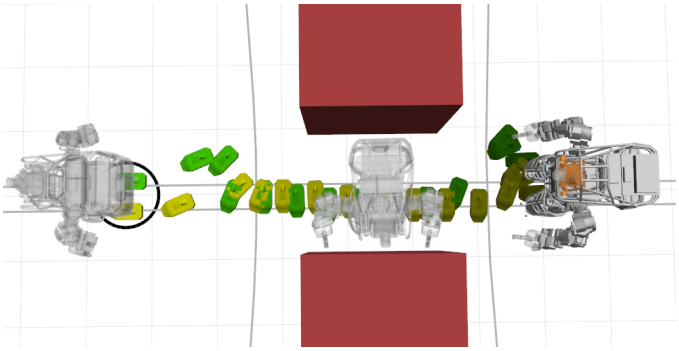


Fig. 5. A footstep plan through a narrow gap, for which Atlas must turn sideways.

steps to known convex safe regions, which we can construct efficiently. The primary advantage of our MIQCQP footstep planner is its ability to generate rich footstep sequences in difficult terrain with guarantees of completeness and global optimality. We do not rely on sampling or fixed action sets, which may miss small regions of safe terrain entirely. Our approach also handles terrain of varying height gracefully, as the same object can be treated as both an obstacle and a walking surface when appropriate, as in Fig. 3.

On the other hand, we are entirely reliant on the existence of sufficiently many convex obstacle-free regions to ensure that a path through the environment can be found. The IRIS algorithm generates these regions efficiently, but currently requires user input to seed the position of each region. This is a mixed blessing: by seeding such a region, the human operator provides valuable input about the possible walking surfaces for the robot, but this requirement clearly sacrifices autonomy of the robot. We are currently investigating ways to fully automate the generation of safe terrain regions. In addition, we require that each safe region represent a planar area of terrain: non-planar terrain regions introduce non-convex constraints in our formulation and cannot be allowed.

In the future, we intend to improve the speed with which we can generate footstep plans, since waiting up to a minute

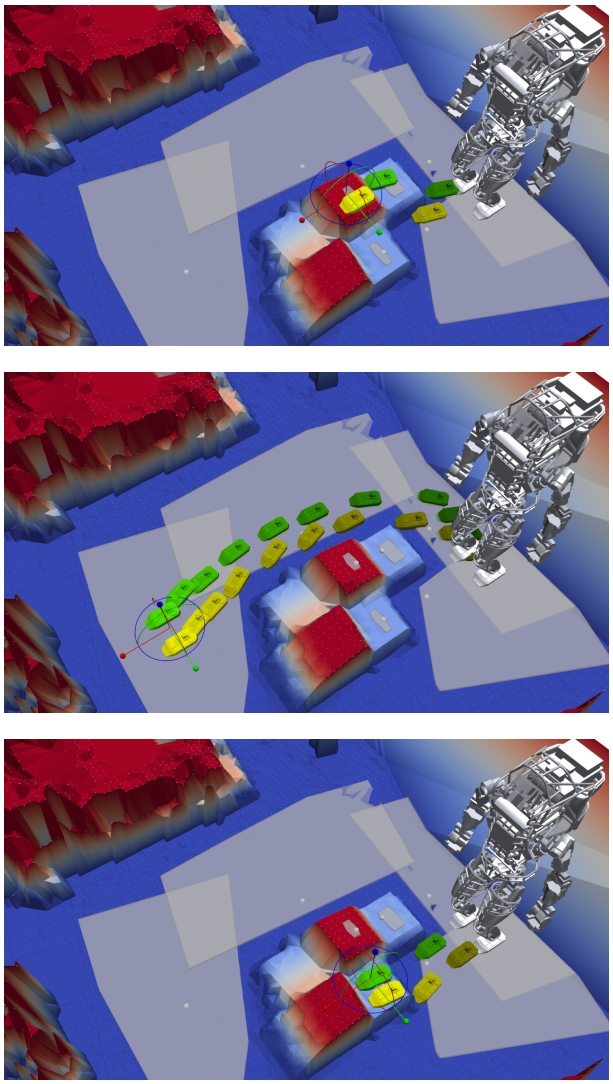


Fig. 6. Footstep plans for to navigate over and around a set of cinderblocks, using data sensed by the Atlas humanoid. Gray shaded areas are the obstacle-free regions of configuration-space, into which the center of each footstep must be placed.

before the robot can start moving may be frustrating in a real-world scenario. We may be able to do this by relaxing the reachability criteria for footsteps which are relatively far in the future. For example, we might consider the reachability and orientation of the first 10 footsteps in a plan, but only ensure that the next 10 or 20 footsteps can be assigned to safe terrain regions without the distance between them becoming too large. This will significantly reduce the complexity of the MIQCQP, at the cost of some likelihood of generating plans through areas that appear feasible under the simplified reachability criteria but through which the robot cannot actually travel.

In addition, we are interested in combining this work with the research on dynamic walking planning currently underway in the Robot Locomotion Group at MIT. Our colleagues intend to use the MIQCQP to produce initial positions and orientations for the footsteps, along with their assignments to safe terrain regions, then run an additional optimization to plan the contact forces on the feet, the center of mass trajectory, and the complete collision-free motions of the robot's limbs. The assignments to convex safe regions produced by the MIQCQP should be a valuable set of convex constraints for this later optimization.

Source Code

Our experimental implementation of the MIQCQP footstep planner in MATLAB is available within Drake, located at `https://github.com/RobotLocomotion/drake`.

ACKNOWLEDGMENTS

The authors are grateful for the support and advice of the members of the Robot Locomotion Group at MIT and Team MIT of the DARPA Robotics Challenge. particularly Pat Marion, Andres Valenzuela, and Hongkai Dai for their expertise, ideas, and enthusiasm.

REFERENCES

[1] R. Tedrake, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2014. [Online]. Available: http://drake.mit.edu

[2] R. L. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Workshop on the Algorithmic Foundations of Robotics*, 2014. [Online]. Available: http://groups.csail.mit.edu/robotics-center/public_papers/Deits14.pdf

[3] P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade, "Vision-guided humanoid footstep planning for dynamic environments," *IEEE-RAS International Conference on Humanoid Robots*, pp. 13–18, 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1573538

[4] L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida, "Real-time replanning using 3d environment for humanoid robot," in *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovnie, 2011, pp. p.584–589. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00601300

[5] J. E. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," in *IEEE-RAS International Conference on Humanoid Robots*, Karlsruhe, Germany, 2003.

[6] J. E. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami, "An adaptive action model for legged navigation planning." in *IEEE-RAS International Conference on Humanoid Robots*, 2007, p. 196202. [Online]. Available: http://www.researchgate.net/publication/224401320_An_adaptive_action_model_for_legged_navigation_planning/file/e0b4951938eaab8795.pdf

[7] J. J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Footstep planning among obstacles for biped robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, Maui, Hawaii, 2001, pp. 500–505.

[8] ——, "Online footstep planning for humanoid robots," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2003, pp. 932–937. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1241712

[9] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake, "Bounding on rough terrain with the LittleDog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 192–215, 2011. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364910388315

[10] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock, "Multi-step motion planning for free-climbing robots," in *Algorithmic Foundations of Robotics VI*, ser. Springer Tracts in Advanced Robotics, M. Erdmann, M. Overmars, D. Hsu, and F. v. d. Stappen, Eds. Springer Berlin Heidelberg, Jan. 2005, no. 17, pp. 59–74. [Online]. Available: http://link.springer.com/chapter/10.1007/10991541_6

[11] P. D. Neuhaus, J. E. Pratt, and M. J. Johnson, "Comprehensive summary of the institute for human and machine cognitions experience with LittleDog," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 216–235, Feb. 2011. [Online]. Available: http://ijr.sagepub.com/content/30/2/216

[12] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, UK; New York: Cambridge University Press, 2004.

[13] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. Perez D'Arpino, R. Deits, M. DiCicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K.-T. Yu, J. Shah, K. Iagnemma, R. Tedrake, and S. Teller, "An architecture for online affordance-based perception and whole-body planning," *Submitted to: Journal of Field Robotics*, 2014. [Online]. Available: http://dspace.mit.edu/handle/1721.1/85690

[14] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic foot step placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010. [Online]. Available: https://groups.csail.mit.edu/robotics-center/elib/papers/Herdt10.pdf

[15] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2014. [Online]. Available: http://www.gurobi.com/

[16] Mosek ApS, "The MOSEK optimization software," 2014. [Online]. Available: http://www.mosek.com/

[17] IBM Corp., "User's manual for CPLEX," 2010. [Online]. Available: http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r2/topic/com.ibm.common.doc/doc/banner.htm

[18] "GNU linear programming kit." [Online]. Available: http://www.gnu.org/software/glpk/glpk.html

[19] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and control of multiple UAVs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*. Monterey, CA: American Institute of Aeronautics and Astronautics, Aug. 2002. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/6.2002-4588

[20] J. Lofberg, "Big-m and convex hulls," 2012. [Online]. Available: http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Tutorials.Big-MAndConvexHulls

[21] MATLAB, *version 8.2.0.701 (R2013b)*. Natick, MA: The MathWorks Inc., 2013.

[22] P.-E. Danielsson and O. Seger, "Generalized and separable sobel operators," in *Machine vision for three-dimensional scenes*, H. Freeman, Ed. Sand Diego, CA: Academic Press, Inc., 1990.