CrossMark

Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot

Scott Kuindersma[1] · Robin Deits[1] · Maurice Fallon[1] · Andrés Valenzuela[1] ·
Hongkai Dai[1] · Frank Permenter[1] · Twan Koolen[1] · Pat Marion[1] · Russ Tedrake[1]

Abstract This paper describes a collection of optimization algorithms for achieving dynamic planning, control, and state estimation for a bipedal robot designed to operate reliably in complex environments. To make challenging locomotion tasks tractable, we describe several novel applications of convex, mixed-integer, and sparse nonlinear optimization to problems ranging from footstep placement to whole-body planning and control. We also present a state estimator formulation that, when combined with our walking controller, permits highly precise execution of extended walking plans over non-flat terrain. We describe our complete system integration and experiments carried out on Atlas, a full-size hydraulic humanoid robot built by Boston Dynamics, Inc.

✉ Scott Kuindersma
  scottk@csail.mit.edu

  Robin Deits
  rdeits@csail.mit.edu

  Maurice Fallon
  mfallon@csail.mit.edu

  Andrés Valenzuela
  avalenzu@csail.mit.edu

  Hongkai Dai
  daih@csail.mit.edu

  Frank Permenter
  fpermenter@csail.mit.edu

  Twan Koolen
  tkoolen@csail.mit.edu

  Pat Marion
  patmarion@csail.mit.edu

  Russ Tedrake
  russt@csail.mit.edu

[1] Computer Science and Artificial Intelligence Laboratory,
   Massachusetts Institute of Technology, Cambridge, MA, USA

1 Introduction

The dream of legged robotics is to achieve reliable, versatile, and dynamic locomotion for a robot capable of doing useful work in a variety of environments. As participants in the DARPA Robotics Challenge (DRC), we are particularly interested in tasks related to disaster relief, such as walking outdoors over irregular terrain and maintaining stability while applying forces to the environment such as when cutting through a wall with a power tool. Disaster scenarios place a premium on the ability to walk over and around obstacles and through narrow passages that require reasoning about the full kinematics of the robot. Several practical challenges arise in the design of these systems, such as how to manage the complexity of the robot and environment model to efficiently do online planning and feedback control and how to achieve sufficiently precise execution given inevitable sensor limitations. In this paper we describe our approach to addressing these problems with Atlas.

Perhaps the most basic capability our system must have is the ability to navigate to a desired location despite the presences of obstacles such as steps, gaps, and debris. Our approach to walking combines an efficient footstep planner with a simple dynamic model of the robot to efficiently compute desired walking trajectories. To plan a sequence of safe footsteps, we decompose the problem into three steps. First, a LIDAR terrain scan is used to identify obstacles in the vicinity of the robot. Given this obstacle map, we solve a sequence of optimization problems to compute a set of convex safe footstep regions in the configuration space of the foot. Next, a mixed-integer convex optimization

problem is solved to find a feasible sequence of footsteps through these regions. Finally, a desired center of pressure trajectory through these steps is computed and input to the controller.

For complex dynamic whole-body motions like climbing out of a car or getting up from the ground, more descriptive kinematic and dynamic models must be used for planning motions. However, for complex humanoid systems like Atlas, solving trajectory optimization problems using the full dynamics can be computationally prohibitive. We describe a direct transcription algorithm (Sect. 3.2) that computes dynamically-feasible trajectories using the full kinematics and centroidal dynamics of the robot. This formulation offers a significant computational advantage over existing full dynamic trajectory optimization algorithms while still producing dynamically-feasible whole-body trajectories for running and jumping with an Atlas model.

We use time-varying linear quadratic regulator (LQR) design to stabilize trajectories for a simplified dynamic model of the robot (Sect. 4.2). By combining the optimal LQR cost-to-go with the instantaneous dynamic, input, and contact constraints of the full robot inside a quadratic program (QP), we exploit the stabilizing properties of LQR while maintaining the versatility afforded by QP-based control formulations in which whole-body motions can be tracked or constrained in a variety of ways. To implement our controller on a physical system requires that we be able to efficiently compute solutions to the QP at each control step. We describe an efficient active-set algorithm capable of finding solutions in less than 1 millisecond for Atlas (68 states and 28 inputs).

Inputs to the controller are computed by a low-drift state estimator that fuses kinematic, inertial, and LIDAR information (Sect. 5). Despite significant kinematic sensor limitations due to backlash and actuator deflection, our experiments demonstrate a measurable improvement in our ability to estimate the robot's state in a variety of experimental scenarios. We show that the robot is capable of walking over nontrivial terrain while maintaining extremely low drift from the desired footstep trajectory—a critically important capability to navigate efficiently through obstacle-ridden environments.

The paper is organized as follows. In the following section, we give a brief overview of the Atlas hardware. In Sect. 3 we describe our footstep planning and dynamic motion planning algorithms. In Sects. 4 and 5 we describe our control and state estimation formulations, respectively. In Sect. 6 we describe several experiments performed on the physical robot evaluating the state estimation and control algorithms in practice. We also describe recent simulation results of controlled highly dynamic motions that are currently being developed for the robot. Finally, we conclude with a discussion of future work in Sect. 7.

2 Atlas

Atlas is a full-scale, hydraulically-actuated humanoid robot manufactured by Boston Dynamics, Inc. The robot stands approximately 188 cm tall with a total mass of 155 kg (without hands attached). It has 28 actuated degrees of freedom: 6 in each leg and arm, 3 in the back, and 1 neck joint. A tether attached to the robot supplies high-voltage 3-phase power for the on-board electric hydraulic pump, distilled water for cooling, and a 10 Gbps fiber-optic line to support communication between the robot and a field computer that runs our planning, estimation, and control software.

Joint position, velocity, and force measurements are generated at 1000 Hz on the robot computer and transmitted back to a field computer. Joint positions are measured by linear variable differential transformers (LVDTs) mounted on the actuators. There are no joint force sensors, but joint forces are inferred using pressure sensors inside the actuators. In addition to the LVDT sensors, digital encoders mounted on the neck and arm joints give low-noise position and velocity measurements (notably these are not available in the legs). A KVH 1750 inertial measurement unit (IMU) mounted on the pelvis provides highly accurate 6-DOF angular rate and acceleration data used for state estimation (Sect. 5). Two 6-axis load cells are mounted to the wrists, and arrays of four strain gauges, one in each foot, provide 3-axis force-torque sensing.

As illustrated in Fig. 1, the robot is equipped with a Multisense SL sensor head designed by Carnegie Robotics which combines a fixed binocular stereo camera with a Hokuyo UTM-30LX-EW planar LIDAR sensor mounted on a spindle that can rotate at up to 30 RPM. The LIDAR captures 40 scan lines of the environment per second, each contain-
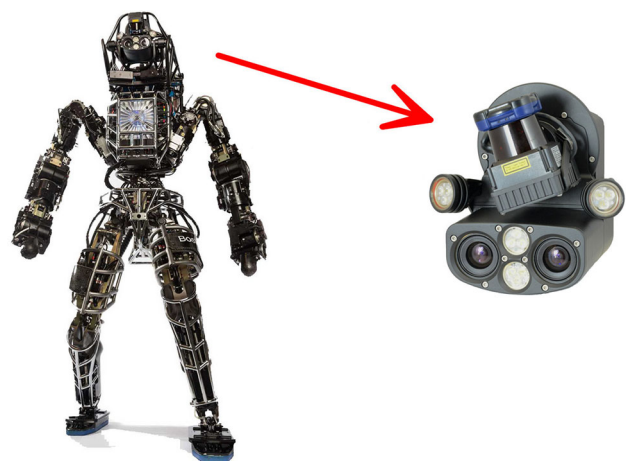

Fig. 1 The Boston Dynamics Atlas humanoid robot and the Carnegie Robotics Multisense SL sensor head equipped with a rotating LIDAR scanner and a stereo cameras. (photo credits: Boston Dynamics and CRL)

ing 1081 range returns out to a maximum range of 30 m. The entire head can pitch up and down but it cannot yaw or roll.

We received the robot on August 12, 2013 and implemented a software system (originally developed in simulation) to compete successfully in the DRC Trials on December 20, 2013. Our related paper describes the many pieces of our software system used in the competition (Fallon et al. 2015). The vast majority of the planning, estimation, and control implementation work described in this paper was done after this event between January and October 2014. In February 2015, Boston Dynamics completed a major upgrade to Atlas that included multiple actuator redesigns, reconfiguration of the arms, 2 additional arm DOFs, approximately 25 kg in additional mass, and capability to operate wirelessly with battery power and onboard computers for perception and control. For simplicity we describe our approach in terms of the earlier system, but we note that the same algorithms have successfully been applied to the most recent version of the robot.

3 Motion planning

Motion planning for legged systems is a fundamentally mixed discrete and continuous optimization problem. Planning algorithms must decide where and when contacts with the environment are initiated or broken and, during periods of unchanging contact, the system must typically move smoothly while maintaining balance and achieving a desired motion or interaction with the environment. For typical walking tasks, we decompose the discrete phase into two parts. We first analyze the environment and compute a set of convex regions where contacts are allowed. Then we solve an optimization problem that assigns contacts to these regions in a way that minimizes cost while respecting kinematic and dynamic constraints.

We will discuss two distinct approaches for assigning contacts to convex regions. The first is for the case of footstep planning, where kinematic constraints on footstep poses are defined with respect to the previous step using approximate reachable regions. This formulation is suitable for the majority of the locomotion scenarios we are interesting in exploring. The second method goes one step further by including the full kinematics and centroidal dynamics of the robot in the optimization to guarantee reachability and support a wider variety of motions and environmental interactions (such as planning to grab handrails or transition from prone to standing), at the expense of increased computation time. We discuss our approach to footstep planning in the following section and our whole-body dynamic planning algorithm in Sect. 3.2.

3.1 Footstep planning as a mixed-integer convex problem

We identify the footstep planning problem as a matter of choosing footstep placements on a given terrain from a start state to a goal while ensuring that the sequence of steps can be safely executed by Atlas. We represent this problem as single mixed-integer convex optimization, in which the number of footsteps and their positions and orientations are simultaneously optimized with respect to some cost function, while ensuring that each footstep is on safe terrain.

Existing footstep planning methods, broadly speaking, fall into two categories: discrete searches and continuous optimizations. We retain some elements from both categories, performing a simultaneous optimization of the discrete assignment of footsteps to convex regions and the continuous position of the footsteps within those regions.

Discrete search approaches have typically made use of a successor set, a list of possible poses for one foot relative to the position of the other foot. From the set of successors, a tree of possible footstep plans can be built and explored to find a path from start to goal. Obstacle avoidance is easily handled by pruning the tree of successors whenever a foot would intersect an obstacle. This approach has been used successfully by Kuffner et al. (2001, 2003), Chestnutt et al. (2003), Michel et al. (2005), and Baudouin et al. (2011). Later work by Chestnutt introduced an adaptive successor set, in which a small continuous search is performed to adjust a step that would result in collision with an obstacle (Chestnutt et al. 2007). That approach was demonstrated with online replanning over rough terrain (Nishiwaki et al. 2011). Shkolnik also used a fixed successor set to represent dynamically feasible bounding motions for a quadruped (Shkolnik et al. 2011). Implementation of these discrete searches requires very careful selection of the successor set: a small set severely restricts the possible motions that can be made, while a large set results in a rapidly branching tree of possible plans. Discrete search techniques such as A^* can be challenging to apply to footstep planning due to the difficulty of finding an informative, admissible heuristic (Garimort and Hornung 2011).

Continuous optimizations avoid the challenges of choosing a particular successor set by allowing the position of each footstep to vary subject to some constraints. The reachability of the robot's legs can be represented with constraints on the relative positions of each footstep, and costs or constraints can be added to ensure that the footstep plan reaches its goal position. If the objective function and constraints are convex, then such an optimization can be solved extremely efficiently (Boyd and Vandenberghe 2004). Herdt performs a convex optimization to plan footstep positions and a center-of-mass trajectory, using linear constraints on the distance from one foot to the next to represent the robot's reachable set (Herdt et al. 2010). However, the need for convex con-

straints prevents this optimization from considering the yaw of the robot or its feet and prevents it from handling obstacle avoidance. Our prior work used a non-convex optimization to find locally optimal footstep plans and was able to include yaw as a decision variable, but could still not effectively avoid obstacles (Fallon et al. 2015).

Prior to planning any walking motion, we classify the area around Atlas as safe or unsafe for footstep placement. In the environments used for the DRC, we found it sufficient to simply exclude areas of the terrain that are steeper than a pre-defined threshold. To plan footstep contacts, we must ensure that each footstep lies in the safe terrain set. Unfortunately, the set of safe terrain is unlikely to be convex or even connected: in an environment as simple as a staircase, the safe terrain consists of the top surface of every step, a non-convex and disconnected set. In order to perform an optimization of footstep placements, we must constrain the footsteps to lie in this non-convex set. In general, when an optimization has non-convex constraints, it can be difficult or impossible to find a globally optimal solution or to prove that none exists (Boyd and Vandenberghe 2004).

Instead, we choose to explicitly represent the combinatorial aspect of footstep planning by decomposing the non-convex set of safe terrain into a set of convex planar safe regions. The approximate decomposition we use is described in Sect. 3.1.1. This transforms the problem of avoiding obstacles into a discrete problem of assigning each footstep to some convex region that is known to be obstacle-free. In principle, the exponential number of possible assignments of footsteps to convex regions may appear to be intractable, but if we restrict our optimization to an objective function that is convex, then it is straightforward to represent the problem of assigning footstep poses to convex regions as a mixed-integer convex problem. In the worst case, this does not eliminate the exponential search through the discrete assignments, but in practice the convex objective can provide an informative heuristic for the search process and dramatically reduce the time needed to find the optimal solution. Excellent tools have been developed in the past decade to solve a variety of mixed-integer convex problems, and the solvers can provide globally optimal solutions or proofs of infeasibility where appropriate (Gurobi Optimization, Inc. 2014; Mosek 2014; IBM Corp. 2010).

The general form of mixed-integer convex programming is

$$
\begin{aligned}
\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad & f(\mathbf{x}, \mathbf{y}) \\
\text{subject to} \quad & \mathbf{g}(\mathbf{x}, \mathbf{y}) \le 0 \\
& \mathbf{y} \in \mathbb{Z}^m,
\end{aligned}
$$

where f and $\mathbf{g}$ are convex functions and the elements of the vector $\mathbf{y} \in \mathbb{Z}^m$ take on integer values. A special case of

this is mixed-(0,1) convex programming, in which the $\mathbf{y}$ are restricted to values of only 0 or 1:

$$
\begin{aligned}
\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad & f(\mathbf{x}, \mathbf{y}) \\
\text{subject to} \quad & \mathbf{g}(\mathbf{x}, \mathbf{y}) \le 0 \\
& \mathbf{y} \in \{0, 1\}^m.
\end{aligned}
$$

We use binary variables of this form to indicate the assignment of footsteps to regions. Let $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N$ be the poses of the footsteps, expressed as position and yaw with $\mathbf{p}_j = (x_j, y_j, z_j, \theta_j)$, and let $G_1, G_2, \ldots, G_R$ be the regions of safe terrain, represented as convex polytopes. We create a matrix $\mathbf{Y} \in \{0, 1\}^{R \times N}$ to represent the assignment of footsteps to safe regions. Our optimization problem is

$$
\begin{aligned}
\underset{\mathbf{p}_1, \ldots, \mathbf{p}_N, Y}{\text{minimize}} \quad & f(\mathbf{p}_1, \ldots, \mathbf{p}_N) \\
\text{subject to} \quad & \mathbf{g}(\mathbf{p}_1, \ldots, \mathbf{p}_N) \le 0 \\
& Y_{i,j} \implies \mathbf{p}_j \in G_i \\
& Y_{i,j} \in \{0, 1\} \\
& \sum_{i=1}^{R} Y_{i,j} = 1 \quad \forall j \in \{1, \ldots N\}.
\end{aligned}
$$

The conditional constraint that $Y_{i,j} \implies \mathbf{p}_j \in G_i$ can be represented exactly using a standard big-M formulation, provided that we have some bounds on the possible values of the $\mathbf{p}_j$ (Bemporad and Morari 1999). Such bounds are easy to provide, since no footstep can be farther from the start pose than the robot's maximum stride length multiplied by the number of footsteps. The additional convex constraints $\mathbf{g}(\mathbf{p}_1, \ldots, \mathbf{p}_N) \le 0$ represent an approximation of the reachable set of footsteps for Atlas, discussed further in Sect. 3.1.2.

3.1.1 Convex decomposition

To simplify the combinatorial problem of assigning footsteps to convex safe regions, we would like to minimize the number of convex pieces into which the safe terrain set is decomposed. This presents a number of challenges. First of all, even for a two-dimensional environment with polygonal obstacles, computing the minimum set of convex obstacle-free pieces that cover the entire environment is computationally very difficult and is known to be NP-hard (Lingas 1982). Secondly, even a truly minimal convex decomposition may result in a very large number of small convex pieces in order to fill in all of the crevices in a cluttered environment (Lien and Amato 2004). Here we sacrifice the notion of covering the entire obstacle-free space and instead focus on creating a few large convex regions. This choice allows us to cover a large fraction of the feasible terrain without creating an unmanageable number of regions.

In order to compute these regions, we have developed IRIS, an algorithm for greedily computing a single large obstacle-free convex region (Deits and Tedrake 2014a). IRIS begins with a seed point that is known to be obstacle-free, provided by our human operator or by a higher-level planner. That seed point forms the center of a very small obstacle-free ellipsoid. The IRIS algorithm alternates between two convex optimizations. In the first step, a series of small quadratic programs are solved to find a set of hyperplanes that separate the ellipsoid from the set of obstacles. Each hyperplane defines an obstacle-free half-space, and the intersection of those half-spaces is a (convex) polytope. In the second step, a single semidefinite program is solved to find the maximum-volume ellipsoid inscribed in that polytope. These two steps can be repeated to grow the ellipsoid until a local fixed point is found. Two complete iterations of the IRIS algorithm are shown in Fig. 2.

The result of IRIS is the final polytope or ellipsoid, either of which can be used as a convex representation of obstacle-free space. We use the polytope representation in our planner, since it is always of larger volume than the (inscribed) ellipsoid and can be represented as a set of linear constraints.

When operating Atlas, our human pilot provides the seed points at which the IRIS algorithm begins. This allows the operator to provide high-level input about which surfaces are appropriate for walking, a task that is well-suited to the pilot's expertise but difficult to perform autonomously. We are also currently investigating methods to automate the selection of seed points, and have demonstrated autonomous seeding of regions using a simple heuristic in a 3D environment (Deits and Tedrake 2015).
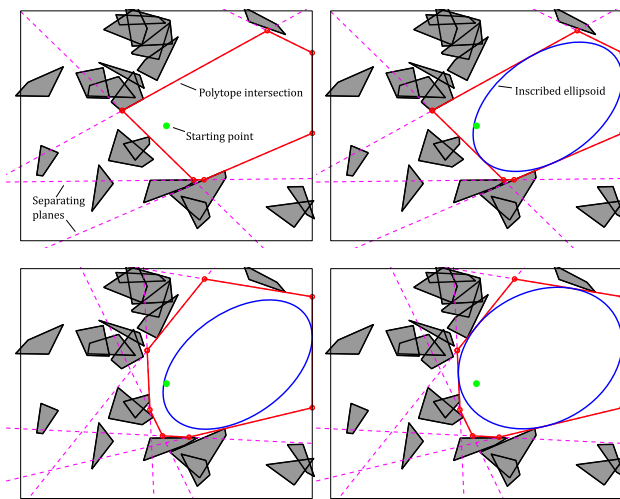
3.1.2 Representing reachability

When planning footstep placements, we must somehow represent the kinematic reachability of the robot, that is, the set of foot placements that can be achieved given the constraints imposed by the dimensions of the limbs and the limits of the joints. Directly reasoning about this reachable set using the full kinematic model of the robot would be ideal, but such reasoning introduces polynomials of trigonometric functions of the robot's joint angles and is not compatible with our convex formulation. Instead, we use a simplified inner approximation of the reachable set for Atlas that can be represented with mixed-integer convex quadratic constraints.

We represent the approximate reachable set of footstep positions as the intersection of circles fixed in the frame of reference of the prior footstep. Each circle has radius d_k and is located at some fixed offset $\mathbf{o}_k$ in the frame of the prior footstep. The reachable region defined by these circles is shown in Fig. 3. For each footstep j, we require that

$$\left\| \begin{bmatrix} x_{j+1} \\ y_{j+1} \end{bmatrix} - \left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} + \begin{bmatrix} \cos\theta_j & -\sin\theta_j \\ \sin\theta_j & \cos\theta_j \end{bmatrix} \mathbf{o}_1 \right) \right\| \le d_1 \qquad (1)$$

$$\left\| \begin{bmatrix} x_{j+1} \\ y_{j+1} \end{bmatrix} - \left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} + \begin{bmatrix} \cos\theta_j & -\sin\theta_j \\ \sin\theta_j & \cos\theta_j \end{bmatrix} \mathbf{o}_2 \right) \right\| \le d_2. \qquad (2)$$

This is not yet a convex constraint, since cos and sin are non-convex functions. To mitigate this, we replace $\sin\theta_j$ and $\cos\theta_j$ with additional decision variables, which we label s_j and c_j, respectively. Equation 1 becomes

Fig. 2 A demonstration of the IRIS algorithm in a planar environment consisting of 20 uniformly randomly placed convex obstacles and a square boundary. Each *row* above shows one complete iteration of the algorithm: on the *left*, the hyperplanes are generated, and their polytope intersection is computed. On the *right*, the *ellipse* is inflated inside the polytope. Figure reproduced from Deits and Tedrake (2014a)
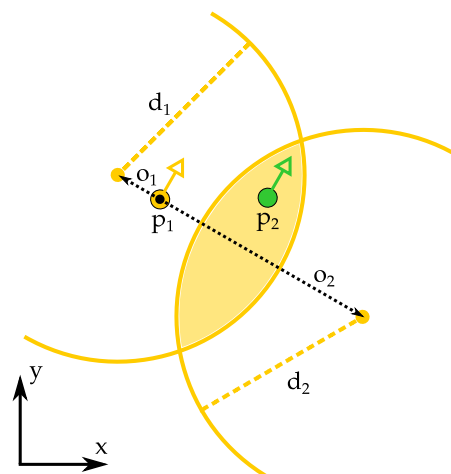

Fig. 3 An approximation of the reachable set of positions for the center of the right foot given the position of the left foot ($\mathbf{p}_1$). The view is from above. The *two circles* have radii d_1 and d_2 and are located at displacements of $\mathbf{o}_1$ and $\mathbf{o}_2$ from $\mathbf{p}_1$, respectively. The *shaded region* shows the set of reachable poses for the center of the right foot in the xy plane. A single feasible pose for the right foot is shown as $\mathbf{p}_2$

$$\left\|\begin{bmatrix} x_{j+1} \\ y_{j+1} \end{bmatrix} - \left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} + \begin{bmatrix} c_j & -s_j \\ s_j & c_j \end{bmatrix}\mathbf{o}_1\right)\right\| \le d_1, \tag{3}$$

which is a convex quadratic constraint. Of course, we still must ensure that s_j and c_j behave like sin and cos. To do this, we create piecewise linear approximations of sin and cos and use additional integer variables to indicate the active linear approximation for a given value of θ_j. By choosing the number of piecewise linear segments that we use in our approximation, we can trade off between accuracy and computational speed. The particular approximation used in our planner is discussed in more detail in Deits and Tedrake (2014b).

Alternatively, we can avoid the non-convexity introduced by sin and cos by fixing the yaw angle of each footstep beforehand, then running the mixed-integer optimization to determine the position and assignment to safe terrain for each footstep. Doing so removes the additional integer variables which were required for our piecewise linear approximation of the unit circle, but requires us to run a second, nonlinear optimization to solve for the yaw angles themselves. We can alternate between the two steps to further refine the footstep plan: using a nonlinear optimization to choose the footstep orientations, then running the mixed-integer convex optimization to choose the optimal position and assignment of those footsteps at the given orientations. The addition of the nonlinear program removes any guarantees of global optimality, but may still produce acceptable footstep plans. We have observed that the nonlinear planner works comparatively well for long footstep plans with few or no obstacles, since it does not require the explicit enumeration of each possible yaw bin for each footstep. When the environment is cluttered, however, the full mixed-integer convex optimization discussed above is required in order to find feasible solutions.

3.1.3 Determining the number of footsteps

In general, we cannot expect to know *a priori* how many footsteps will be needed to reach a target position, so the footstep planner must be responsible for determining this number. Since the entire set of footsteps is simultaneously optimized, changing the number of footsteps alters the size of the optimization problem. We can, of course, simply try a variety of numbers of footsteps, performing a separate optimization each time, but this results in a great deal of wasted computation. Instead, we add a binary flag to each footstep to indicate that the particular step is unused. We label this flag ρ_j and require that if ρ_j is true, then footstep j be fixed to the starting pose of that foot

$$\rho_j \implies \mathbf{p}_j = \begin{cases} \mathbf{p}_1 & \text{if } j \text{ is odd} \\ \mathbf{p}_2 & \text{if } j \text{ is even,} \end{cases} \tag{4}$$
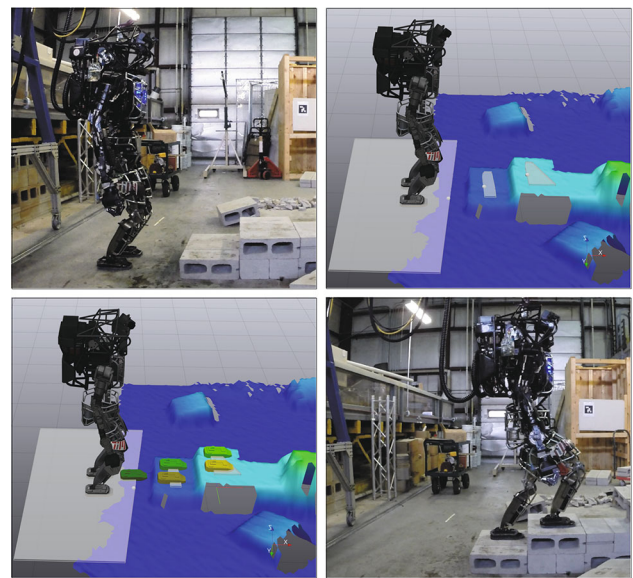

Fig. 4 Illustration of the footstep planner being used to climb cinder blocks. *Top* the robot approaches cinder block steps, the user selects clicks regions of interest on the terrain map, and convex regions are optimized using IRIS. *Bottom* the robot plans footsteps through the convex regions using mixed-integer optimization and the robot executes the desired footsteps

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the fixed initial poses of the feet. Adding a negative cost on each ρ_j to the objective in our optimization allows us to reward the planner for taking fewer footsteps without knowing beforehand how many will be required. After the optimization is complete, any footstep with ρ_j equal to 1 can be removed from the plan.

To complete the formulation, we add linear constraints on the change in z and yaw values from one footstep to the next, which prevents the robot from turning or climbing too far in a single step. We also add a quadratic cost on the displacement between adjacent footsteps, to penalize very large steps that may be more likely to cause a fall. The result is a single mixed-integer quadratically-constrained quadratic program (MIQCQP) that, when solved to optimality, chooses the position and orientation of each footstep, the total number of footsteps to take, and the assignment of those footsteps to convex safe regions of terrain. Figure 4 illustrates the use of this planner in practice.

We formulate the entire footstep planning optimization as follows:

$$\underset{\mathbf{p}_1,\dots,\mathbf{p}_N,\mathbf{S},\mathbf{C},\mathbf{Y},\rho_1,\dots,\rho_N}{\text{minimize}} (\mathbf{p}_N - \mathbf{p}_g)^\top \mathbf{Q}_g(\mathbf{p}_N - \mathbf{p}_g) + \sum_{j=1}^{N} q_\rho \rho_j$$

$$+ \sum_{j=1}^{N-1} (\mathbf{p}_{j+1} - \mathbf{p}_j)^\top \mathbf{Q}_r(\mathbf{p}_{j+1} - \mathbf{p}_j))$$

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

safe terrain regions:

$$Y_{r,j} \implies \mathbf{A}_r \mathbf{p}_j \leq \mathbf{b}_r \qquad\qquad r = 1, \ldots, R$$

piecewise linear $\sin\theta$:

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

piecewise linear $\cos\theta$:

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

approximate reachability:

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

fix extra steps to initial pose:

$$\rho_j \implies \mathbf{p}_j = \begin{cases} \mathbf{p}_1 & \text{if } j \text{ is odd} \\ \mathbf{p}_2 & \text{if } j \text{ is even.} \end{cases}$$

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

$$Y_{r,j}, S_{\ell,j}, C_{\ell,j}, \rho_j \in \{0, 1\}$$

bounds on step positions and differences:

$$\mathbf{p}_{\min} \leq \mathbf{p}_j \leq \mathbf{p}_{\max}$$
$$\Delta\mathbf{p}_{\min} \leq (\mathbf{p}_j - \mathbf{p}_{j-1}) \leq \Delta\mathbf{p}_{\max}$$

where $\mathbf{p}_g \in \mathbb{R}^4$ is the x, y, z, θ goal pose, $\mathbf{Q}_g \in \mathbb{S}_+^4$ and $\mathbf{Q}_r \in \mathbb{S}_+^4$ 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 $\mathbf{p}_{\min}, \mathbf{p}_{\max}, \Delta\mathbf{p}_{\min}, \Delta\mathbf{p}_{\max} \in \mathbb{R}^4$ are bounds on the absolute footstep positions and their differences, respectively. We also fix $\mathbf{p}_1$ and $\mathbf{p}_2$ to the initial poses of the robot's feet. The $g_{s,\ell}, h_{s,\ell}$ and $g_{c,\ell}, h_{c,\ell}$ terms represent a pre-selected piecewise linearization of the sine and cosine functions, described in more detail in Deits and Tedrake (2014b).

Despite the very large number of discrete decisions involved in solving the footstep planning problem to optimality, the mixed-integer convex formulation leads to extremely efficient solutions in typical cases. For a footstep plan of $N = 12$ steps, in which each step must be assigned to one of $R = 10$ safe regions, $L = 8$ piecewise linearizations of sin and cos, and 2 values of each ρ_j, there are, naively, $10^{12} \times 8^{12} \times 2^{12} \approx 3 \times 10^{26}$ possible discrete combinations to explore. However, the convex objective and constraints allow the solver to avoid exploring the vast majority of that search space without sacrificing optimality. In Deits and Tedrake (2014b) we demonstrate typical solve times of 1 to 10 seconds for problems of such a size. When the entire terrain is flat and no safe region assignments are needed, solve times are typically less than 1 second for plans of up to 16 footsteps.

Given a desired footstep trajectory, a dynamic walking motion can be defined using a piecewise polynomial center of pressure (COP) trajectory through the footsteps (this idea is described further in Sect. 4). However, for more complex, whole-body motions like running or getting up from the ground, we require the richer planning formulation described next.

3.2 Dynamic motion planning

For humanoid robot performing complex motions in nontrivial environments, kinematic and dynamic constraints often appear together. For example, a robot jumping down off a ledge must reason about the contact forces being applied during launch, its center of mass (COM) velocity at the point of takeoff, the position of its foot with respect to the ledge during flight, and the kinematic reachability of its legs during landing.

The descriptiveness of the dynamic model used for planning strongly affects the range of possible motions. At one end of the spectrum are optimizations that reason about the full hybrid dynamics of the legged system. Such approaches have been shown to produce beautiful results in model systems (Mombaur 2009; Posa et al. 2014), but they remain computationally expensive for high-dimensional systems like Atlas. At the other end are methods based on reduced dynamical models, where assumptions about the local flatness of terrain and absence of angular momentum greatly simplify the dynamics. To produce a larger variety of dynamic multi-contact motions, we have developed an approach that strikes a balance between these extremes and plans using the full kinematics of the robot to enforce geometric contact conditions and a dynamic model that encodes the relationship between the contact force on the robot and the robot's total linear and angular momenta.
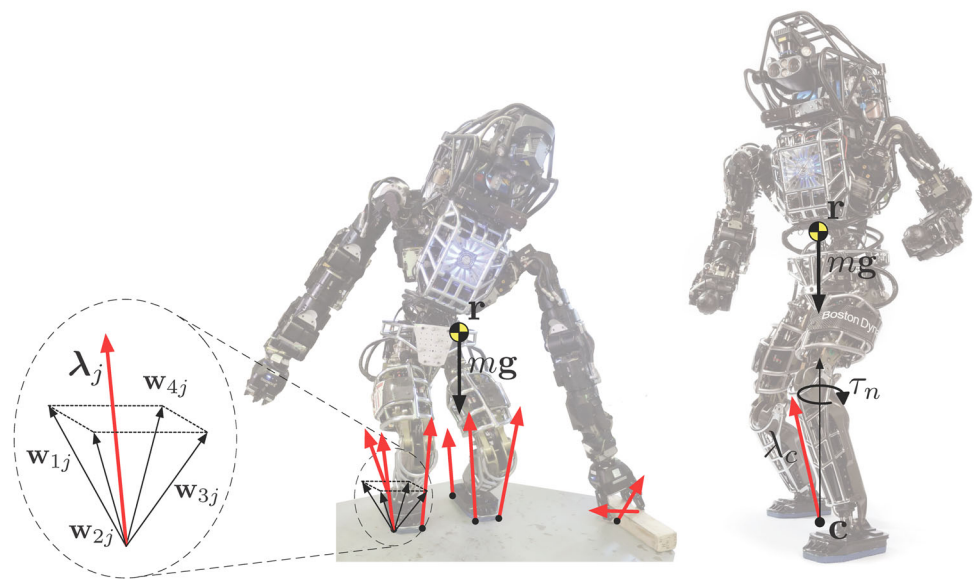
For all mechanical systems, the rate of the total linear and angular momentum of the system must equal the net external wrench (force/torque) on the system. The instantaneous momenta are functions of the generalized position, $\mathbf{q}$, and velocity, $\mathbf{v}$. In our implementation we represent the floating base positions and velocities using Euler angles and their derivatives, although singularity-free representations can also be used. The *centroidal momentum* is the total momentum defined in a coordinate frame at the instantaneous center of mass (COM) position and aligned with the world frame. Orin and Goswami (2008) showed that it can be computed easily using the centroidal momentum matrix, $\mathbf{A}_G$:

$$\mathbf{h}_G = \mathbf{A}_G(\mathbf{q})\mathbf{v} \equiv \begin{bmatrix} \mathbf{k} \\ \mathbf{l} \end{bmatrix}, \tag{5}$$

where $\mathbf{k}$ and $\mathbf{l}$ are the centroidal angular and linear momenta, respectively.

The rate of the centroidal momentum can be expressed by differentiating (5) or by writing it as a function of external forces (Lee and Goswami 2012; Dai et al. 2014):

Fig. 5 *Left* A conservative friction polyhedron with $N_d = 4$. The force vector, $\boldsymbol{\lambda}_j$, at contact j is a positive combination of the generating vectors, $\mathbf{w}_{ij}$, $i = 1, \ldots, 4$. *Center* external forces acting at a set of contact points (6). *Right* external forces and torques acting on the robot summarized at the COP (20)

$$\dot{\mathbf{k}} = \sum_{j=1}^{N_c} (\mathbf{c}_j - \mathbf{r}) \times \boldsymbol{\lambda}_j$$

$$\dot{\mathbf{l}} = m\ddot{\mathbf{r}} = m\mathbf{g} + \sum_{j=1}^{N_c} \boldsymbol{\lambda}_j, \tag{6}$$

where $\mathbf{r}$ is the COM of the robot, m is the total mass, $\mathbf{g}$ is the acceleration due to gravity, and $\mathbf{c}_j$, $\boldsymbol{\lambda}_j$, are the contact position and force, respectively, at point j. N_c is the number of active contact points. As will be discussed below, we use a redundant multiple-force description of the total wrench acting on a rigid body because it permits the use of simple linear friction constraints in our optimization. There has been compelling recent work in controlling the momenta of humanoids for balance and locomotion (Koyanagi et al. 2008; Lee and Goswami 2012; Herzog et al. 2013; Koolen et al. 2013a), including the resolved momentum control framework proposed by Kajita et al. (2003) and Neo et al. (2007). Inspired by this work, we design our planning algorithm using the concept of centroidal momentum.

Our approach is to plan trajectories using the full kinematics while reasoning about external forces and moments and using the centroidal dynamics to ensure dynamic feasibility. In doing so, we are making a fundamental assumption that the robot's $n - 6$ internal degrees of freedom are fully actuated, and that the dynamic constraints can therefore be defined in terms of the 6 floating base DOFs. This results in an optimization problem with far fewer decision variables and constraints compared to an optimization that includes the full dynamics. In addition, the gradients of the constraints (which are required by most nonlinear optimization problem solvers) are significantly easier to compute. The price for this reduced computation is the inability to reason about internal torques. However, in practice this is not a tremendous limitation given

the considerable strength of Atlas' actuators. Even for robots with very restrictive actuator limits, there may be some computational benefit to using the output of this optimization as an initial seed for a trajectory optimization problem that reasons about the full dynamics.

To generate reasonable dynamic motions, friction constraints on the contact forces must be satisfied. We use a standard, conservative polyhedral approximation of the friction cone for each active contact point, $\mathbf{c}_j$, and require that

$$\boldsymbol{\lambda}_j = \sum_{i=1}^{N_d} \beta_{ij}\mathbf{w}_{ij}, \quad \beta_{ij} \geq 0. \tag{7}$$

The generating vectors, $\mathbf{w}_{ij}$, are computed as $\mathbf{w}_{ij} = \mathbf{n}_j + \mu_j \mathbf{d}_{ij}$, where $\mathbf{n}_j$ and $\mathbf{d}_{ij}$ are the contact-surface normal and i^{th} tangent vector for the jth contact point, respectively, μ_j is the Coulomb friction coefficient, and N_d is the number of tangent vectors used in the approximation (Pollard and Reitsma 2001). Figure 5 illustrates this idea graphically.

Trajectory optimization algorithms fall into two general classes: shooting methods and transcription methods (Betts 1998). Shooting methods involve only the control inputs as decision variables and must simulate the system forward in order to evaluate the cost function. Transcription methods, on the other hand, include a finite set of states along the trajectory as decision variables and incorporate the dynamics of the system as constraints on the state and input variables. By simultaneously optimizing the states and inputs along the trajectory, transcription methods avoid numerical issues that can be present in shooting methods (in which small changes in inputs early in the trajectory can lead to large changes in cost and final state) and avoid the need to simulate the system dynamics during optimization.

We formulate a direct transcription problem by representing the differential constraints as algebraic equations at $k = 1, \ldots, M$ points, with $h[k]$ denoting the time interval between knot point k and $k + 1$:

$$\underset{\Gamma}{\text{minimize}} \quad \sum_{k=1}^{M} L(\mathbf{q}[k], \mathbf{v}[k], \ddot{\mathbf{r}}[k], \boldsymbol{\lambda}[k], h[k])$$

$$\text{subject to} \quad m\ddot{\mathbf{r}}[k] = m\mathbf{g} + \sum_j \boldsymbol{\lambda}_j[k] \qquad \text{(linear momentum)}$$

$$\mathbf{k}[k] = \mathbf{A}_G^{\mathbf{k}}(\mathbf{q}[k])\mathbf{v}[k] \qquad \text{(angular momentum)}$$

$$\dot{\mathbf{k}}[k] = \sum_j (\mathbf{c}_j[k] - \mathbf{r}[k]) \times \boldsymbol{\lambda}_j \qquad \text{(angular momentum rate)}$$

$$\forall_j \ \boldsymbol{\lambda}_j[k] = \sum_{i=1}^{N_d} \beta_{ij}[k]\mathbf{w}_{ij} \qquad \text{(friction)}$$

$$\forall_{i,j} \ \beta_{ij}[k] \geq 0$$

$$\mathbf{r}[k] = COM(\mathbf{q}[k]) \qquad \text{(COM location)}$$

Kinematic constraints

Time integration constraints

where the set of decision variables, Γ, is

$$\Gamma = \{\mathbf{q}[k], \mathbf{v}[k], h[k], \mathbf{r}[k], \dot{\mathbf{r}}[k], \ddot{\mathbf{r}}[k], \mathbf{c}_j[k], \boldsymbol{\lambda}[k], \boldsymbol{\beta}[k],$$
$$\mathbf{k}[k], \dot{\mathbf{k}}[k] \mid k = 1, \ldots, M\}, \qquad (8)$$

and the cost function, $L(\mathbf{q}[k], \mathbf{v}[k], \ddot{\mathbf{r}}[k], \boldsymbol{\lambda}[k], h[k])$, is equal to

$$h[k] \left(\|\mathbf{q}[k] - \mathbf{q}_{\text{nom}}[k]\|_{\mathbf{Q}_q}^2 + \|\mathbf{v}[k]\|_{\mathbf{Q}_v}^2 \right.$$
$$\left. + \|\ddot{\mathbf{r}}[k]\|^2 + \sum_j \|\boldsymbol{\lambda}_j\|_{\mathbf{Q}_\lambda}^2 \right), \qquad (9)$$

where $\|\mathbf{x}\|_{\mathbf{Q}}^2$ is shorthand for the quadratic cost, $\mathbf{x}^T \mathbf{Q} \mathbf{x}$, $\mathbf{Q} \succeq 0$, and $\mathbf{A}_G^{\mathbf{k}}$ is a matrix formed by taking the top three rows of $\mathbf{A}_G$. The cost term ($\|\mathbf{q}[k] - \mathbf{q}_{\text{nom}}[k]\|_{\mathbf{Q}_q}^2$) regularizes the configuration of the robot to a nominal pose (e.g., a typical standing posture).

The time integration constraints include backward-Euler integration of the generalized velocities, rate of angular momentum and COM acceleration, and mid-point integration on COM velocity.

$$h[k]\mathbf{v}[k] = \mathbf{q}[k] - \mathbf{q}[k-1] \qquad (10)$$

$$h[k]\dot{\mathbf{k}}[k] = \mathbf{k}[k] - \mathbf{k}[k-1] \qquad (11)$$

$$h[k]\ddot{\mathbf{r}}[k] = \dot{\mathbf{r}}[k] - \dot{\mathbf{r}}[k-1] \qquad (12)$$

$$\frac{h[k]}{2}(\dot{\mathbf{r}}[k] + \dot{\mathbf{r}}[k-1]) = \mathbf{r}[k] - \mathbf{r}[k-1], \qquad (13)$$

The kinematic constraints typically include unilateral joint limits, contact location, and collision-free constraints. Similar constraint sets have been employed to control whole-body humanoid motions previously (Saab et al. 2011; Dalibard et al. 2013). Dalibard et al. (2013) separately solve the kinematic planning problem using full body model and a point mass dynamic model. Our approach combines these two problems together with a more descriptive dynamic model, and is able to generate motions that cannot be handled by this kind of two-stage planner (such as the running and jumping examples in Sect. 6.4).

The gradients of this optimization problem are sparse, since most of the constraints only depend on decision variables at one or two knot points. Nonlinear programs with sparse gradients can be solved efficiently using powerful sequential quadratic program (SQP) solvers like SNOPT (Gill et al. 2005). Solution times vary with the particular planning problem; for the examples shown in Sect. 6.4 our un-optimized implementation took between 1 and 10 min to find solutions on machines with 3.1–3.3 GHz Intel i7 CPUs. More detail including examples of this approach being applied to plan quadruped gaits and complex humanoid trajectories like traversing monkey bars is available (Dai et al. 2014).

4 Controller design

Our approach to feedback control can be summarized in the following way. Given a planned trajectory for a reduced model of the dynamics (e.g., centroidal dynamics), we com-

pute a time-varying linearization around the trajectory and derive a stabilizing controller using time-varying linear-quadratic regulator (TV-LQR) design. But rather than using the closed-form optimal controller from LQR, we formulate and solve a quadratic program (QP) that additionally captures the instantaneous dynamics, input, and contact constraints of the full walking system. In the following sections, we write the problem in a general form first, then describe particular implementations used in conjunction with the planners described previously.

4.1 General formulation

Given the dynamics, $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, and a desired trajectory, $\mathbf{x}_d(t), \mathbf{u}_d(t)$ for $t \in [0, t_f]$, we linearize along the trajectory,

$$\dot{\bar{\mathbf{x}}}(t) = \mathbf{A}(t)\bar{\mathbf{x}}(t) + \mathbf{B}(t)\bar{\mathbf{u}}(t), \tag{14}$$

using the relative coordinates, $\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_d(t)$ and $\bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_d(t)$. Next we define a quadratic cost function,

$$g(\mathbf{x}(t), \mathbf{u}(t), t) = \bar{\mathbf{x}}^T(t)\mathbf{Q}\bar{\mathbf{x}}(t) + \bar{\mathbf{u}}^T(t)\mathbf{R}\bar{\mathbf{u}}(t), \tag{15}$$

and solve the constrained minimization problem,

$$\begin{aligned} \underset{\mathbf{u}(t)}{\text{minimize}} \quad & \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt \\ \text{subject to} \quad & \dot{\bar{\mathbf{x}}}(t) = \mathbf{A}(t)\bar{\mathbf{x}}(t) + \mathbf{B}(t)\bar{\mathbf{u}}(t) \\ & \mathbf{Q} = \mathbf{Q}^T \succeq 0 \\ & \mathbf{R} = \mathbf{R}^T \succ 0. \end{aligned}$$

The solution to this problem is given by solving the continuous Riccati differential equation, which yields the optimal cost-to-go (Tedrake 2014b),

$$J(\mathbf{x}, t) = \bar{\mathbf{x}}^T(t)\mathbf{S}(t)\bar{\mathbf{x}}(t) + \bar{\mathbf{x}}^T(t)\mathbf{s}_1(t) + s_2(t). \tag{16}$$

By the Hamilton–Jacobi–Bellman (HJB) equation (Bertsekas 1995), we know that the optimal controller satisfies

$$\bar{\mathbf{u}}^*(t) = \arg\min_{\bar{\mathbf{u}}} \ell(\bar{\mathbf{x}}, \bar{\mathbf{u}}, t), \tag{17}$$

$$\ell(\bar{\mathbf{x}}, \bar{\mathbf{u}}, t) \equiv \bar{\mathbf{u}}^T(t)\mathbf{R}\bar{\mathbf{u}}(t) + \left(2\bar{\mathbf{x}}^T(t)\mathbf{S}(t) + \mathbf{s}_1^T(t)\right)\mathbf{B}\bar{\mathbf{u}}(t), \tag{18}$$

for all $\bar{\mathbf{x}}$, where we have dropped constant terms from the minimization. It is easy to show that (17) can be written as a time-varying linear policy simply by taking the derivative of $\ell(\bar{\mathbf{x}}, \bar{\mathbf{u}}, t)$ with respect to $\bar{\mathbf{u}}$, setting it equal to zero, and then solving for $\bar{\mathbf{u}}$.

The key point here is that for physical systems we typically have inequality constraints that cannot be ignored, such as

torque limits or constraints on ground reaction forces. As long as these constraints can *instantaneously* be expressed as linear inequalities, we can formulate and solve a convex[1] QP to compute inputs at each control step given $\bar{\mathbf{x}}$,

$$\underset{\bar{\mathbf{u}}}{\text{minimize}} \, \ell(\bar{\mathbf{x}}, \bar{\mathbf{u}}, t) \quad \text{subject to} \quad \mathbf{M}\bar{\mathbf{u}} \le \mathbf{b}. \tag{19}$$

In other words, this optimization finds the steepest descent direction of the cost-to-go subject to the constraints of the system. Note that inputs computed by solving this QP are, in general, not equal to those computed by thresholding the output of the closed-form LQR policy. In addition, as we will see below, using this framework allows for additional constraints to be added to the QP with little additional computational cost.

4.2 COM and COP stabilization

Consider the case where our walking plan is in the form of a piecewise polynomial center of pressure (COP) trajectory, $\mathbf{c}_d(t), t \in [0, t_f]$, derived based on a desired footstep plan. In our current implementation, $\mathbf{c}_d(t)$ is a piecewise linear trajectory that interpolates between the planned footstep centroids with timing governed by high-level walking parameters such as swing speed and double-support time.

For a legged system on locally flat terrain, the centroidal dynamics (6) can be rewritten in terms of the COP (Lee and Goswami 2012), $\mathbf{c}$, as

$$\begin{aligned} \dot{\mathbf{k}} &= (\mathbf{c} - \mathbf{r}) \times \boldsymbol{\lambda}_c + \boldsymbol{\tau}_n, \\ \dot{\mathbf{l}} &= m\mathbf{g} + \boldsymbol{\lambda}_c, \end{aligned} \tag{20}$$

where $\mathbf{c}$ is the COP, $\boldsymbol{\lambda}_c$ is the net external force at the COP, and $\boldsymbol{\tau}_n$ is the normal contact moment. This is illustrated on the right side of Fig. 5. If we assume that the centroidal angular momentum of the robot, $\dot{\mathbf{k}} = 0$, $\mathbf{k} = 0$, and the normal moment, $\boldsymbol{\tau}_n = 0$, the centroidal dynamics simplify to

$$\begin{aligned} \dot{\mathbf{x}}_{\text{CM}} &= \mathbf{A}\mathbf{x}_{\text{CM}} + \mathbf{B}\mathbf{u}_{\text{CM}} \equiv \begin{bmatrix} 0_{2\times2} & \mathbf{I}_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix} \mathbf{x}_{\text{CM}} \\ &\quad + \begin{bmatrix} 0_{2\times2} \\ \mathbf{I}_{2\times2} \end{bmatrix} \mathbf{u}_{\text{CM}}, \end{aligned} \tag{21}$$

$$\mathbf{c} = \begin{bmatrix} \mathbf{I}_{2\times2} & 0_{2\times4} \end{bmatrix} \mathbf{x}_{\text{CM}} - \frac{r_z}{\ddot{r}_z + g} \mathbf{I}_{2\times2} \mathbf{u}_{\text{CM}}, \tag{22}$$

where $\mathbf{x}_{\text{CM}} = [r_x, r_y, \dot{r}_x, \dot{r}_y]^T$ is the ground projection of the COM, and $\mathbf{u}_{\text{CM}} = [\ddot{r}_x, \ddot{r}_y]^T$.

The outputs (22) are nonlinear in general, but they are linear time-varying given a twice differentiable COM height trajectory, or linear time-invariant if we assume the COM

---

[1] Note that $\ell(\bar{\mathbf{x}}, \bar{\mathbf{u}}, t)$ is convex since $\mathbf{R} \succ 0$.

height remains constant (resulting in the well-studied linear inverted pendulum dynamics). In practice, this is often a reasonable assumption to make despite violations that inevitably occur during execution. In our implementation, we fix the COM height and use the linear form of the outputs (22).

We formulate the LQR problem,

$$
\begin{aligned}
\underset{\mathbf{u}_{\mathrm{CM}}(t)}{\text{minimize}} \quad & \int_0^\infty g(\mathbf{x}_{\mathrm{CM}}(t), \mathbf{u}_{\mathrm{CM}}(t), t)\,dt \\
\text{subject to} \quad & \dot{\bar{\mathbf{x}}}_{\mathrm{CM}}(t) = \mathbf{A}\bar{\mathbf{x}}_{\mathrm{CM}}(t) + \mathbf{B}\bar{\mathbf{u}}_{\mathrm{CM}}(t) \\
& \mathbf{c}(t) = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{0}_{2\times4} \end{bmatrix}\mathbf{x}_{\mathrm{CM}}(t) - \frac{r_z}{g}\mathbf{I}\mathbf{u}_{\mathrm{CM}}(t) \\
& \mathbf{c}_d(t) = \mathbf{c}_d(t_f), \quad \forall t \geq t_f \\
& \mathbf{Q} = \mathbf{Q}^T \succ 0 \\
& \mathbf{R} = \mathbf{R}^T \succ 0.
\end{aligned}
$$

where $\bar{\mathbf{x}}_{\mathrm{CM}} = \mathbf{x}_{\mathrm{CM}}(t) - [\mathbf{c}_d^T(t_f), 0, 0]^T$ and $\bar{\mathbf{u}}_{\mathrm{CM}}(t) = \mathbf{u}_{\mathrm{CM}}(t)$. Here the cost is defined in terms of outputs $\bar{\mathbf{c}}(t) = \mathbf{c}(t) - \mathbf{c}_d(t)$,

$$
g(\mathbf{x}_{\mathrm{CM}}(t), \mathbf{u}_{\mathrm{CM}}(t), t) = \bar{\mathbf{c}}^T(t)\mathbf{Q}_c\bar{\mathbf{c}}(t) + \bar{\mathbf{u}}_{\mathrm{CM}}^T(t)\mathbf{R}\bar{\mathbf{u}}_{\mathrm{CM}}(t), \tag{23}
$$

where the right hand side can be written in terms of $\mathbf{x}_{\mathrm{CM}}(t)$, $\bar{\mathbf{u}}_{\mathrm{CM}}(t)$, and $\mathbf{c}_d(t)$ by substituting in the linear output equation. Deriving the Riccati equation for this problem reveals that $\mathbf{S}$ has no time-dependent terms (amounting to the continuous algebraic Riccati equation). The linear and affine cost-to-go coefficients then become linear differential equations that we can solve with a single backward pass along the trajectory (i.e. without having to numerically integrate). This means that in practice $\ell(\bar{\mathbf{x}}_{\mathrm{CM}}, \bar{\mathbf{u}}_{\mathrm{CM}}, t)$ can be computed very efficiently on time scales appropriate for receding-horizon footstep planning. A paper containing a full description of this algorithm is currently in preparation.

4.3 Stabilizing whole-body plans

The planning algorithm described in Sect. 3.2 provides richer inputs to the controller in the form of full kinematic, centroidal momentum, and external force trajectories. We therefore have more flexibility in how we design our LQR problem. However, in keeping with the simplicity of the COP LQR formulation, we stabilize the 3D COM trajectory using simple linear COM dynamics. Given the COM trajectory $\mathbf{r}_{\mathrm{des}}(t), \dot{\mathbf{r}}_{\mathrm{des}}(t), \ddot{\mathbf{r}}_{\mathrm{des}}(t)$, we define a quadratic cost,

$$
g(\mathbf{x}_{\mathrm{CM}}(t), \mathbf{u}_{\mathrm{CM}}(t), t) = \bar{\mathbf{x}}_{\mathrm{CM}}^T(t)\mathbf{Q}\bar{\mathbf{x}}_{\mathrm{CM}}(t) + \bar{\mathbf{u}}_{\mathrm{CM}}^T(t)\mathbf{R}\bar{\mathbf{u}}_{\mathrm{CM}}(t), \tag{24}
$$

where $\bar{\mathbf{x}}_{\mathrm{CM}}(t) = \mathbf{x}_{\mathrm{CM}}(t) - [\mathbf{r}_{\mathrm{des}}^T(t), \dot{\mathbf{r}}_{\mathrm{des}}^T(t)]^T$, $\bar{\mathbf{u}}_{\mathrm{CM}}(t) = \mathbf{u}_{\mathrm{CM}}(t) - \ddot{\mathbf{r}}_{\mathrm{des}}(t)$, and the dynamics are taken to be the

the 3D analog of (21). We found empirically in simulation experiments that adding terms to explicitly control angular momentum in addition to the COM trajectory did not noticeably improve stability.

If the motion of the robot involves a flight phase, then we must modify the LQR problem to account for the fact that the robot COM motion is completely determined by gravity. To track a desired motion in flight the robot must achieve the desired COM state at the time of take off. Thus the robot COM dynamics becomes hybrid: it has the smooth double integrator dynamics above in the support phases, and a discrete transition from takeoff to landing state in the flight phase. We compute the cost-to-go for this hybrid system using the jump Riccati equation (Manchester et al. 2011). Intuitively, this hybrid LQR approach simply encodes the goal of tracking the the COM trajectory up to the point of takeoff and after landing.

4.4 Additional costs and constraints

In practice, the QP formulation (19) must be augmented with additional cost terms and linear constraints to achieve satisfactory performance. To begin with, since the LQR dynamics we derived above are not the full robot dynamics, we must add additional constraints to our QP to compute feasible inputs to the robot. Given the current robot state, $\mathbf{q}, \mathbf{v}$, we can compute the equations of motion,

$$
\mathbf{H}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{C}(\mathbf{q}, \mathbf{v}) = \mathbf{B}\boldsymbol{\tau} + \mathbf{J}^T\boldsymbol{\lambda}, \tag{25}
$$

$$
\begin{bmatrix} \mathbf{H}_f \\ \mathbf{H}_a \end{bmatrix}\dot{\mathbf{v}} + \begin{bmatrix} \mathbf{C}_f \\ \mathbf{C}_a \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_a \end{bmatrix}\boldsymbol{\tau} + \begin{bmatrix} \mathbf{J}_f^T \\ \mathbf{J}_a^T \end{bmatrix}\boldsymbol{\lambda}, \tag{26}
$$

where $\mathbf{H}(\mathbf{q})$ is the mass matrix, the vector $\mathbf{C}(\mathbf{q}, \mathbf{v})$ captures the gravitational and Coriolis terms, $\mathbf{B}$ is the control input map, and $\mathbf{J}^T$ transforms external forces, $\boldsymbol{\lambda}$, into generalized forces. We have explicitly partitioned the dynamics into unactuated and actuated DOFs. For typical humanoids like Atlas, the only unactuated DOFs are the floating base and $\mathbf{B}_a$ is full rank.

We can use this decomposition to eliminate the input torques, $\boldsymbol{\tau}$, as a decision variable and represent torque constraints as linear inequality constraints on $\boldsymbol{\lambda}$ and $\dot{\mathbf{v}}$. For the case of robot walking, the vector $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1^T \ldots \boldsymbol{\lambda}_{N_c}^T]^T$ contains ground contact forces acting at N_c contact points. We use the same polyhedral approximation described in Sect. 3.2 (with $N_d = 4$) to ensure the contact forces remain inside the friction cone.

In addition to respecting friction constraints, the controller must detect when it can assign nonzero values to external force variables. For example, during or when entering a swing phase, the QP should not assign positive ground reaction forces to that foot. For this we used a simple logic to determine what contact force variables should be included in the

optimization: if the robot expects to be in contact (based on the footstep plan) *or* foot strain gages are reporting significant values on the foot, the force variables for that foot can be assigned positive values. The exception to this is when the foot is breaking contact during walking, where the plan is used exclusively to determine whether a foot is in contact or not. In practice, if the planned contact state is active but no force is measured, we set the friction coefficient to be a small constant value (effectively only allowing normal contact forces on that body). This helps to reduce unwanted tangential motions of the foot prior to contact.

Since the LQR solutions do not capture the whole-body motion of the robot, we specify additional motion goals as desired accelerations of a set of frames attached to bodies on the robot, such as the feet and pelvis. Desired accelerations of a body frame, $\ddot{\mathbf{p}}(t)$, at time t are computed using a PD rule,

$$\ddot{\mathbf{p}}_{\text{ref}}(t) = \mathbf{K}_p(\mathbf{p}_d(t) - \mathbf{p}(t)) + \mathbf{K}_d(\dot{\mathbf{p}}_d(t) - \dot{\mathbf{p}}(t)) + \ddot{\mathbf{p}}_d(t), \quad (27)$$

where $\mathbf{p}_d(t)$ is a smooth twice-differentiable desired trajectory for the frame. For typical walking, smooth foot trajectories are computed by the footstep planner and desired pelvis motion is computed from the footstep plan. In our implementation, the desired pelvis yaw is equal to the average foot yaw, and the pelvis height is maintained at a constant height above the feet. For whole-body dynamic motions, we track the smooth foot, pelvis, torso, and hand trajectories specified by the plan.

In addition to or in place of body accelerations, desired generalized accelerations, $\dot{\mathbf{v}}_d$, can be input and incorporated as costs or constraints in the QP. For example, generalized acceleration constraints can be useful for maintaining a fixed upper body configuration while walking and carrying an object. In practice we found it extremely helpful to quickly be able to try different combinations of constraints and change constraints to costs (and vice versa). Our freely-available implementation in Drake (2014a) supports a variety of options for making these changes with minimal effort.

The complete QP used in our experiments described in Sect. 6 is described in Quadratic Program 1.

Quadratic Program 1

$$\begin{aligned}
\underset{\dot{\mathbf{v}}, \lambda, \beta, \varepsilon}{\text{minimize}} \quad & \ell(\bar{\mathbf{x}}_{\text{CM}}, \bar{\mathbf{u}}_{\text{CM}}, t) \\
& + \sum_{i \notin \mathcal{C}} w_i \|\ddot{\mathbf{p}}_i(\dot{\mathbf{v}}) - \ddot{\mathbf{p}}_{i,\text{ref}}(t)\|^2 + w_\varepsilon \|\varepsilon\|^2
\end{aligned}$$

subject to
$$\mathbf{H}_f \dot{\mathbf{v}} - \mathbf{J}_f^T \lambda = -\mathbf{C}_f \qquad \text{(dynamics)}$$

$$\forall_{i \in \mathcal{C}} \quad \ddot{\mathbf{p}}_i(\dot{\mathbf{v}}) = -\eta \dot{\mathbf{p}}(t) + \varepsilon \qquad \text{(no slip)}$$

$$|\varepsilon| \leq \varepsilon_{max}$$

$$\forall_{j=\{1...N_c\}} \quad \lambda_j = \sum_{i=1}^{N_d} \beta_{ij} \mathbf{w}_{ij} \qquad \text{(friction)}$$

$$\forall_{i,j} \quad \beta_{ij} \geq 0$$

$$\mathbf{B}_a^{-1}(\mathbf{H}_a \dot{\mathbf{v}} + \mathbf{C}_a - \mathbf{J}_a^T \lambda) \in [\boldsymbol{\tau}_{\min}, \boldsymbol{\tau}_{\max}] \qquad \text{(input)}$$

Here the set $\mathcal{C}$ contains the indices of bodies that are in contact with the environment. For bodies in contact, we apply a "no slip" constraint that, for example, can require that the acceleration of contact points be 0 or some nonzero value in the direction opposite their velocity. To avoid infeasibility, we incorporate slack variables, ε, that permit bounded violations of these constraints (subject to additional cost). For Atlas balancing in double support, the QP has 90 decision variables, 30 equality constraints, and 112 inequality constraints. Note that $\bar{\mathbf{x}}_{\text{CM}}$ and $\bar{\mathbf{u}}_{\text{CM}}$ can be expressed as affine functions of the decision variables given the state of the robot, $(\mathbf{q}, \mathbf{v})$:

$$\bar{\mathbf{x}}_{\text{CM}} = \begin{bmatrix} \text{COM}(\mathbf{q}) \\ \mathbf{J}_{\text{CM}}(\mathbf{q})\mathbf{v} \end{bmatrix} - \bar{\mathbf{x}}_{\text{CM,des}} \qquad (28)$$

$$\bar{\mathbf{u}}_{\text{CM}} = \dot{\mathbf{J}}_{\text{CM}}(\mathbf{q})\mathbf{v} + \mathbf{J}_{\text{CM}}(\mathbf{q})\dot{\mathbf{v}} - \bar{\mathbf{u}}_{\text{CM,des}}, \qquad (29)$$

where $\mathbf{J}_{\text{CM}}(\mathbf{q})$ is the COM Jacobian.

Several researchers have recently explored using QPs to control bipedal systems both in simulation (Abe et al. 2007; Collette et al. 2007; Macchietto et al. 2009; Kudoh et al. 2002; Koolen et al. 2013b; Feng et al. 2013) and in hardware (Ames 2012; Herzog et al. 2013; Saab et al. 2013; Johnson et al. 2015). As in our formulation, these optimizations typically employ low-dimensional dynamic quantities such as center of mass (COM) acceleration (Abe et al. 2007), rate of change of linear and angular momenta (Macchietto et al. 2009; Herzog et al. 2013; Koolen et al. 2013b), zero moment point (ZMP) (Saab et al. 2013), or canonical walking functions (Ames 2012), where the QP cost consists of a weighted distance from a reference value computed using a simple PD control rule (Abe et al. 2007; Macchietto et al. 2009; Herzog et al. 2013).

Johnson et al. (2015) implemented a QP-based momentum controller to achieve balancing and walking with an Atlas robot and compete successfully in the 2013 DRC Trials. Herzog et al. (2013, 2014) used a QP controller for balance control in a hydraulic humanoid lower-body. Our approach shares several features with these approaches, particularly in the definition of constraints, but it differs in the use of time-varying LQR to stabilize trajectories and construct QP objectives.

Our use of the optimal cost-to-go as an objective creates a connection with control Lyapunov techniques. Ames et al. (2012), Ames (2013) used control Lyapunov functions for walking by solving QPs that minimize the input norm, $\|\mathbf{u}\|$, while satisfying constraints on the negativity of $\ell(\mathbf{x}, \mathbf{u}, t)$. In the discrete time setting, Wang and Boyd (2011) describe an approach to quickly evaluating control Lyapunov policies using explicit enumeration of active sets in cases where the number of states is roughly equal to the square of the number of inputs.

4.5 Efficient QP solver

Implementation on hardware demands that sufficiently high control rates be achieved. In our case, that means that we must be able to formulate and solve Quadratic Program 1 in a short amount of time. A key observation is that, during typically operation, the *active set* of inequality constraints changes very infrequently between consecutive control steps. We can exploit this by using an efficient active-set solver. We describe the algorithm briefly below, for more information including timing results against several general-purpose QP solvers, please see our previous paper (Kuindersma et al. 2014).

Quadratic Program 1 can be written in the standard form,

$$
\begin{aligned}
\underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{z}^T\mathbf{W}\mathbf{z} + \mathbf{g}^T\mathbf{z} \\
\text{subject to} \quad & \mathbf{A}\mathbf{z} = \mathbf{b} \\
& \mathbf{M}\mathbf{z} \leq \mathbf{f}
\end{aligned}
\tag{30}
$$

where the inequalities are defined by $\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_n)^T$ and $\mathbf{f} = (f_1, \ldots, f_n)^T$. To solve this problem, it is assumed that $\mathbf{m}_i^T\mathbf{z} = f_i$ at optimality for each i in a subset $\mathcal{A} \subseteq \{1 \ldots n\}$ called the *active set*. For time step $k > 0$, this subset equals the indices of the active inequalities from time step $k - 1$. With this assumption, the KKT conditions for the QP can be written in terms of $\mathbf{z}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\alpha}$:

$$
\begin{aligned}
-\mathbf{g} &= \mathbf{W}\mathbf{z} + \mathbf{A}^T\boldsymbol{\alpha} + \sum_{i \in \mathcal{A}} \gamma_i \mathbf{m}_i \\
\mathbf{A}\mathbf{z} &= \mathbf{b} \\
\mathbf{m}_i^T\mathbf{z} &= f_i \quad \forall i \in \mathcal{A} \\
\gamma_i &= 0 \quad \forall i \neq \mathcal{A}
\end{aligned}
\tag{31}
$$

$$
\begin{aligned}
\mathbf{M}\mathbf{z} &\leq \mathbf{f} \\
\gamma_i &\geq 0 \quad \forall i \in \mathcal{A}.
\end{aligned}
\tag{32}
$$

Our method solves the linear equations (31) and checks if the solution $(\mathbf{z}, \boldsymbol{\gamma}, \boldsymbol{\alpha})$ satisfies the inequalities (32). If the inequalities are satisfied, $\mathbf{z}$ solves the QP and the algorithm terminates. Otherwise, the algorithm adds index i to $\mathcal{A}$ if $\mathbf{m}_i^T\mathbf{z} > f_i$ or removes index i if $\gamma_i < 0$ and resolves (31). The algorithm repeats this process until the inequalities (32) are satisfied or a until a specified maximum number of iterations is reached. The method is outlined in Algorithm 1.

On rare occasions when no solution is found, the algorithm fails over to a more reliable (but on average slower) interior point solver. This contingency is required since finite termination cannot be guaranteed for this algorithm. In practice, instances of QP 1 are almost always solved in one iteration. Solving the QP for Atlas during typical walking takes approximately 0.2 ms (1 ms including QP setup time) (Kuindersma et al. 2014). Including all additional controller software components, such as those that evaluate the footstep trajectories, determine whether a body is in contact, handle messages to and from the robot etc., the controller runs at a rate of approximately 800 Hz.

Data: QP of the form (30), active set $\mathcal{A}$
Result: Optimal solution $\mathbf{z}$ with active set $\mathcal{A}$ or Failure.

1   $iter \leftarrow 0$
2   **repeat**
3   Compute candidate solution $\mathbf{z}, \boldsymbol{\gamma}, \boldsymbol{\alpha}$ from (31)
4   **if** $\mathbf{m}_i^T\mathbf{z} > f_i$ **then**
5   add i to $\mathcal{A}$
6   **end**
7   **if** $\gamma_i < 0$ **then**
8   remove i from $\mathcal{A}$
9   **end**
10    $iter \leftarrow iter + 1$
11   **if** $iter > iter_{\text{MAX}}$ **then**
12   **return** Failure
13   **end**
14 **until** $\mathbf{z}$ and $\boldsymbol{\gamma}$ satisfy (32)
15 **return** $\mathcal{A}$ and $\mathbf{z}$

Algorithm 1: Active-set method for solving convex QPs. The set $\mathcal{A}$ passed to the algorithm at time step k equals the set of constraints active at optimality for time $k - 1$.

Other uses of active-set methods for model-predictive control (MPC) have exploited the temporal relationship between the QPs arising in MPC. Bartlett et al. compared active-set and interior-point strategies for MPC (Bartlett et al. 2000). The described an active-set approach based on Schur complements for efficiently resolving KKT conditions after changes are made to the active set. Ferreau et al. (2008) consider the MPC problems where the cost function and dynamic constraints are the same at each time step; i.e., the QPs solved at iteration differ only by a single constraint that enforces initial conditions. By smoothly varying the initial conditions from the previous to the current state, they were able to track a piecewise linear path traced by the optimal solution, where knot points in the path correspond to changes in the active set. Since the controller we considered had changing cost and constraint structure, this method would have been difficult to apply.

4.6 Joint-level control

Transitioning from simulation to hardware requires good low-level control that tracks references from the QP by commanding the robot's hydraulic actuators. The joint-level loop on the robot runs at 1000 Hz and computes servo valve commands, $m \in [-10, 10]$ mA. For lower-body control, we have two approaches that we selectively implement based on the operating situation and state of the robot. The first uses torque-only feedback and the second combines torque and velocity feedback.

When compliant interaction with the terrain is desired, we do torque-only control in the two ankle joints. For example when the slope of the ground cannot be accurately estimated, or in the presence of small obstacles like boards or rocks, we exploit the natural compliance of the ankles to conform to the terrain under the foot. For torque-only control, we combined feedback gains on the sensed torque at the joint (computed via actuator pressure measurements mapped through a fixed transmission curve) with *feedforward* velocity gains to cancel the actuator dynamics. We found that, with the addition of simple piecewise-linear friction models fit from data, compliant control was possible, but the ability to precisely position limbs was limited due to model errors and sensor limitations (e.g., the lack of joint torque sensors).

Alternatively, by using combined torque and velocity feedback, we were able to get very good position tracking results when used with inverse-dynamics-based controllers, but at the expense of reducing natural compliance of the joint. For this mode, the valve servo command has the form,

$$m = K_p^\tau (\tau_{\text{ref}} - \tau) + K_p^v (v_{\text{ref}} - v). \qquad (33)$$

To compute v_{ref}, we output the generalized accelerations in addition to the joint torques solved for by the QP block and integrate them over time:

$$v_{\text{ref}}(T) = \int_{t_c}^{T} \dot{v}_{\text{ref}}(t)dt, \qquad (34)$$

where t_c is the last time the contact state changed (i.e. integrated velocities are reset when feet make and break contact with the terrain).

5 State estimation

The controller requires a high-rate, low-latency estimate of the full state of the robot at every control step. This section presents a state estimator which fulfills this need, has low drift, and satisfies the computation constraints required to run on-line. Figure 6 illustrates the flow of signals from planners and the state estimator to the controller. The core filtering approach was presented in our previous paper (Fallon et al. 2014) and was adapted from the estimator introduced by Bry et al. (2012).

5.1 Requirements and approach

The sensors that are available for use in state estimation comprise an accurate inertial measurement unit (IMU) attached to the pelvis, joint position sensors at each joint, as well as exteroceptive sensors: LIDAR and vision, the latter of which is currently not used.
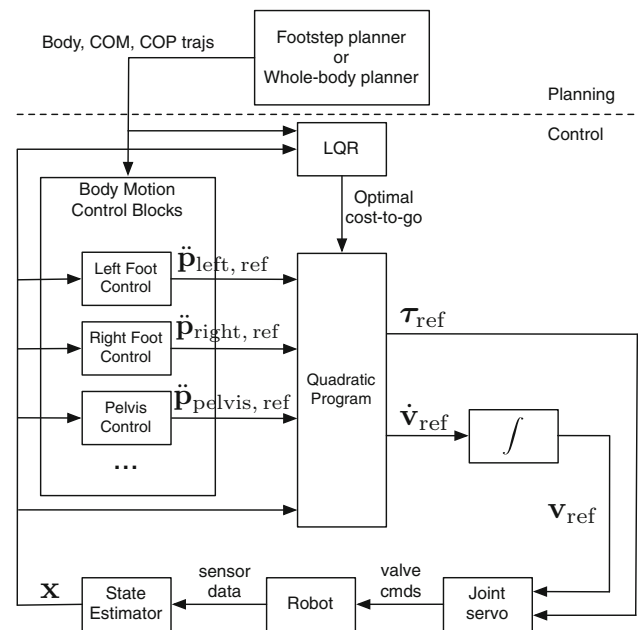

Fig. 6 Block diagram illustrating the flow of signals through the system. The footstep planner (Sect. 3.1) or the whole-body motion planner (Sect. 3.2) provide input desired trajectories to the control system. The controller (Sect. 4) runs in a closed loop with the state estimator (Sect. 5) at approximately 800 Hz. LQR solutions can be recomputed online (typically in a separate thread) using the current state of the robot to reduce the systems sensitivity to deviations from the nominal walking trajectory

The state estimator uses these sensors to produce estimates of the position and velocity of the revolute joints, as well as the pose and twist of the 'floating base' (i.e. the pelvis link). Since the robot has no joint velocity sensors, velocity estimates must be derived from position differencing and filtering. Moreover, while high-quality measurement of the floating base orientation can be readily achieved with an IMU, achieving high-precision positioning with low drift remains a significant challenge. To allow traversal of the uneven terrain described in Sect. 3.1, a drift rate below 1 cm per step is required. The proposed state estimator runs at 333 Hz with a latency of 0.5 ms for the floating base state estimate.

Rather than estimating the full state of the robot using a single process model, we factor the problem by estimating the joint states separately from the floating base. We filter the leg joint positions and velocities using a simple first-order Kalman filter for each joint. The filtered leg joint positions are subsequently used to estimate the state of the floating base. We estimate the floating-base state using an extended Kalman filter (EKF).

To support longer duration plan execution, we remove global drift by localizing to the robot's environment. In Sect. 5.5.2 we discuss how this can be achieved by using the LIDAR sensor to localize against a map of the robot's environment.

5.2 Related work

There are several notable examples of state estimators being developed for legged systems in the literature. Stephens (2011) demonstrated state estimation of a humanoid using a dynamics model and the planned state trajectories. Xinjilefu et al. (2014) extended this approach and applied it to the Atlas robot. They avoid computational challenges of formulating a single extended Kalman filter (EKF) for a humanoid with many degrees of freedom, and propose instead to estimate the pelvis position and joint dynamics in separate filters.

An EKF-based estimator is presented by Bloesch et al. (2012) for a quadruped that uses a prediction model driven by inertial measurements and creates filter corrections using foothold measurements. This approach incorporates the positions of footholds into the state vector (using a point model for each foot) and analyzed system consistency and observability. This approach has also been extended to bipedal locomotion on a simulated version of the SARCOS humanoid robot (Rotella et al. 2014).

While significant progress has been made in robotic mapping using combinations of vision, LIDAR and other sensors, its usage on a humanoid platform in field operations has been limited due to computation, latency and sensitivity reasons. An example result by Stasse et al. (2006) demonstrated loop-closing with a real-time monocular vision SLAM system in a laboratory setting.

Finally, the work of Hornung et al. (2010) utilized a LIDAR sensor to localize an Aldebaran NAO robot within a multi-level environment in a manner similar to our algorithm presented in Sect. 5.5.2. However that approach was limited in accuracy and probabilistic consistency as estimation was carried out with a first-order particle filter separate from the estimation of velocities and the robot's height.

5.3 Signal preprocessing

Some preprocessing is required to obtain useful measurement data. This preprocessing step is specific to the Atlas robot.

Firstly, we note that the IMU measures vibrations induced by the hydraulic pressurizer, which corrupt the acceleration and rotation rate measurements. We notch filter these signals to remove this 85 Hz signal before integration. Secondly, the angle of each leg joint is sensed by measuring the travel of its hydraulic actuator with an LVDT, and then computing a transformation through the joint linkage. This mapping does not account for flexion of the joint linkage when loaded or backlash when a joint changes direction. To account for these effects, we preprocess the joint angle measurements using the correction model, previously introduced by Johnson et al. (2015). The model assumes linear compliance at the joint level,

$$q_i^{\text{preproc}} = q_i^{\text{raw}} - \tau_i / K_i \tag{35}$$

where q_i^{raw} is the raw joint angle measurement, τ_i is the measured joint torque, and K_i is the joint-level stiffness. In practice, we limit the magnitude of the modification term, $|\tau_i/K_i|$, to 0.1 rad. We used $K_i = 10000$ Nm/rad for all joints except the hip yaw joints, where we used $K_i = 7000$ (Johnson et al. 2015).

5.4 Process model

5.4.1 State

The state of the floating base can be described in singularity-free fashion by the tuple

$$\mathbf{x}_{\text{base}} = \left({}^b\boldsymbol{\omega}_b, {}^b\mathbf{v}_b, {}^w\mathbf{R}_b, {}^w\mathbf{p}_b \right), \tag{36}$$

where ${}^b\boldsymbol{\omega}_b \in \mathbb{R}^3$ and ${}^b\mathbf{v}_b \in \mathbb{R}^3$ are the angular and linear velocities, ${}^w\mathbf{R}_b$ is the floating-base rotation matrix, and ${}^w\mathbf{p}_b = [p_x, p_y, p_z]^T$ is the base position. The angular and linear velocities are expressed in the base (pelvis) frame b, while the position and orientation of the pelvis are expressed in a fixed world frame w (as indicated by the leading superscripts above).

As the IMU provides accurate measurements of the angular velocity ${}^b\boldsymbol{\omega}_b$, we use the measured values directly and do not incorporate ${}^b\boldsymbol{\omega}_b$ in the EKF state. In addition to the floating-base state, the EKF also estimates gyro and acceleration biases, denoted $\mathbf{b}_\omega$ and $\mathbf{b}_a$ respectively. This leads us to define the *global* state of the EKF as the tuple

$$\mathbf{x}_{\text{base}} = \left({}^b\mathbf{v}_b, {}^w\mathbf{R}_b, {}^w\mathbf{p}_b, \mathbf{b}_\omega, \mathbf{b}_a \right) \tag{37}$$

The fact that the rotation matrix ${}^w\mathbf{R}_b$ is an overparameterization of orientation leads to constraints on the global state that cannot be handled in a standard EKF setting. To overcome this problem, we maintain orientation uncertainty in terms of a perturbation rotation in body frame represented by a minimal set of coordinates.

Omitting superscripts and subscripts and summarizing (Bry et al. 2012), the true orientation of the floating base is represented by the rotation matrix $\mathbf{R}$. The estimated orientation, $\hat{\mathbf{R}}$, is related to it through

$$\mathbf{R} = \hat{\mathbf{R}}\mathbf{R}(\chi), \tag{38}$$

where $\chi \in \mathbb{R}^3$ is the rotation error in exponential coordinates relative to the body frame and $\mathbf{R}(\chi) = e^{\hat{\chi}}$ is the corresponding perturbation rotation matrix. Here, $\hat{\cdot}$ denotes the skew-symmetric matrix such that $\hat{\cdot}x = \cdot \times x$ for any $x \in \mathbb{R}^3$,

so that $e^{\hat{\chi}}$ is the rotation matrix corresponding to a rotation of $|\chi|$ about the axis defined by χ.

Parameterizing the orientation error in terms of a minimal set of coordinates $\chi \in \mathbb{R}^3$ allows the orientation error covariance to be maintained as a 3×3 matrix $\boldsymbol{\Sigma}_\chi$. This covariance matrix has no inherent constraints which would appear if an overparameterization of orientation were used, and allows easy interpretation. However, we still maintain the mean estimate of the floating base orientation, $\hat{\mathbf{R}}$, as a rotation matrix to avoid singularity issues inherent to any three-degree-of-freedom orientation parameterization.

We can now define the *error* state vector of the EKF as

$$
\mathbf{e}_{\text{base}} = \begin{pmatrix} {}^b\tilde{\mathbf{v}}_b \\ \chi \\ {}^w\tilde{\mathbf{p}}_b \\ \tilde{\mathbf{b}}_\omega \\ \tilde{\mathbf{b}}_a \end{pmatrix}, \tag{39}
$$

where $\tilde{\cdot}$ is the estimated value of $\cdot$ minus the actual value. Covariances are maintained in terms of this error state vector.

5.4.2 Inputs

Similar to angular velocity, the acceleration ${}^b\mathbf{a}_b$ is measured accurately by the IMU. We treat both ${}^b\boldsymbol{\omega}_b$ and ${}^b\mathbf{a}_b$ as inputs to the process model rather than measurements in the EKF – a common approach. Hence, the input vector for the EKF can be written as

$$
\mathbf{u}_{\text{base}} = \begin{bmatrix} {}^b\boldsymbol{\omega}_b \\ {}^b\mathbf{a}_b \end{bmatrix}. \tag{40}
$$

5.4.3 Dynamics

We separately define the dynamics of the global state, used to propagate the mean of the state estimate, and those of the error state, used to propagate the covariances.

The evolution of the global floating-base state can be derived from the Newton–Euler equations. The gyro and acceleration bias dynamics are modeled by a simple random walk. This results in the following dynamics for the global state:

$$
\begin{aligned}
{}^b\dot{\mathbf{v}}_b &= -{}^b\bar{\boldsymbol{\omega}}_b \times {}^b\mathbf{v}_b + {}^w\mathbf{R}_b^T {}^w\mathbf{g} + {}^b\bar{\mathbf{a}}_b + \mathbf{w}_v \\
{}^w\dot{\mathbf{R}}_b &= {}^w\mathbf{R}_b {}^b\hat{\bar{\boldsymbol{\omega}}}_b \\
{}^w\dot{\mathbf{p}}_b &= {}^w\mathbf{R}_b {}^b\mathbf{v}_b + \mathbf{w}_p \\
\dot{\mathbf{b}}_\omega &= \mathbf{w}_{b_\omega} \\
\dot{\mathbf{b}}_a &= \mathbf{w}_{b_a},
\end{aligned} \tag{41}
$$

where ${}^w\mathbf{g}$ denotes gravitational acceleration in world frame; $\mathbf{w}_v$, $\mathbf{w}_p$, $\mathbf{w}_{b_\omega}$ and $\mathbf{w}_{b_a}$ are Gaussian random variables; and

${}^b\bar{\boldsymbol{\omega}}_b$ and ${}^b\bar{\mathbf{a}}_b$ are the bias-corrected angular velocity and linear acceleration vectors, respectively:

$$
{}^b\bar{\boldsymbol{\omega}}_b = {}^b\boldsymbol{\omega}_b + \mathbf{b}_\omega \tag{42}
$$

$$
{}^b\bar{\mathbf{a}}_b = {}^b\mathbf{a}_b + \mathbf{b}_a. \tag{43}
$$

The error dynamics can be trivially derived from the global state dynamics for all but the orientation error, which is expressed in terms of the exponential coordinates χ. Bortz (1971) showed that the orientation error dynamics can be written as

$$
\begin{aligned}
\dot{\chi} = {}^b\boldsymbol{\omega}_b &+ \frac{\chi \times {}^b\boldsymbol{\omega}_b}{2} \\
&+ \frac{1}{\|\chi\|^2} \left[1 - \frac{\|\chi\| \sin \|\chi\|}{2(1 - \cos \|\chi\|)} \right] \chi \times \left(\chi \times {}^b\boldsymbol{\omega}_b \right).
\end{aligned} \tag{44}
$$

Following the linearization derivations from Bry et al. (2012), Euler integration is used to produce the *a priori* state and covariance.

5.5 Measurement model

Here we describe how each sensing modality is used to form Kalman measurement updates of parts of the state vector—as summarized in Table 1. Section 5.5.1 describes the measurements due to leg kinematics, and Sect. 5.5.2 describes the positioning information derived from the LIDAR sensor.

5.5.1 Leg kinematics

As with many floating base state estimation algorithms (Stephens 2011; Bloesch et al. 2012), our approach to using the leg kinematics assumes that the robot's stance foot maintains stationary contact with the ground during part of the gait. This allows instantaneous velocity and position measurements of the robot's floating base to be inferred via forward kinematics. Given the accuracy of the IMU orientation sensors, we choose to use joint sensing to measure

Table 1 Contribution of various sensors to the filtered state estimate

Quantity	Pos $^w\mathbf{p}_b$	Orient $^w\mathbf{R}_b$	Velocity $^b\mathbf{v}_b$	Ang rate $^b\boldsymbol{\omega}_b$	Accel $^w\dot{\mathbf{v}}_b$
Accelerometers					✓
Gyroscopes				✓	
Leg sensing	✗	✗	✓	✗	
LIDAR	✓	✓			

Modes of integration found to be useful are marked ✓ and those not used here (for a variety of reasons) are indicated ✗

only the position and velocity of the pelvis. Of course, perfectly clean and stable ground contact is seldom achieved in practice. However, we assert that for short periods (on the order of the sample time of our sensors) these assumptions are reasonable.

An estimate $^{w}\hat{\mathbf{p}}_b[k]$ of the position of the floating base at time step k can be computed using the current floating base orientation estimate in the world frame, $^{w}\mathbf{R}_b[k]$, and the previous estimated position of the stationary foot, $^{w}\mathbf{p}_f[k-1]$,

$$^{w}\hat{\mathbf{p}}_b[k] = {}^{w}\mathbf{p}_f[k-1] + {}^{w}\mathbf{p}_{b/f}[k], \tag{45}$$

where $^{w}\mathbf{p}_{b/f}[k]$ denotes the position of the floating base with respect to the foot, which can be computed using the filtered leg joint positions and the floating base orientation estimate, $^{w}\boldsymbol{\theta}_b[k]$.

Two types of filter measurement could be formulated using this position estimate. The simplest approach would be to directly apply (45) as a position measurement within the EKF. However, because of the inaccuracy in joint sensing and because the robot's foot does not always remain motionless after initial contact, we use the difference between consecutive position estimates over a short period of time to create a velocity measurement of the pelvis frame,

$$^{w}\hat{\mathbf{v}}_b[k] = \frac{{}^{w}\mathbf{p}_b[k] - {}^{w}\mathbf{p}_b[k-1]}{h}, \tag{46}$$

where $h = 3\,\mathrm{ms}$ is the time step duration. This approach is more attractive because each resultant observation is a separate measurement of the robot's velocity and does not accumulate a history of, for example, the effects of non-ideal ground contact or the footplate rolling or sliding. The influence of an erroneous velocity is transient and quickly corrected by subsequent observations. Using both measurement types together has been explored in related work (Xinjilefu et al. 2014), but we avoid doing so as it raises the possibility of creating inconsistencies, particularly when combined with position measurements derived from the LIDAR module (presented in the following section).

To use the leg kinematics for state estimation, it is necessary to determine which of the feet is most likely to be in stationary contact with the ground. We use a Schmitt trigger with a threshold of 575 N to classify contact forces sensed by the robot's 3-axis foot force-torque sensors and detect whether either foot is in contact. A simple state machine then decides which foot is most reliably in contact and thus will provide the basis for kinematic measurements. The output of the foot contact classifier is demonstrated in the upper plot in Fig. 7.

In the specific case of walking up stairs, the controller needs to use the toe of the trailing foot to push the robot forward and upward while the foot is not in stationary contact. While the design of the state estimator is almost entirely
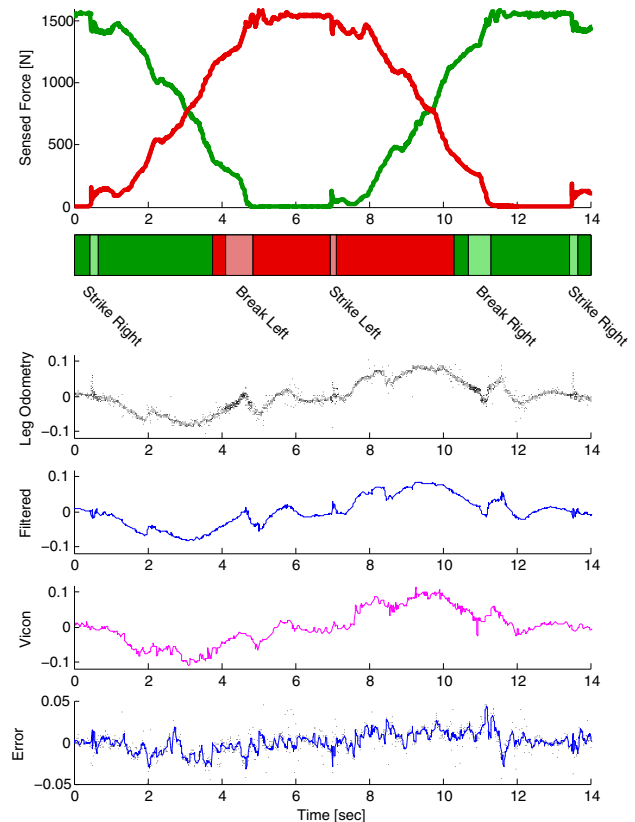

Fig. 7 *Top* Evolution of foot force signals for the left (*green*) and right (*red*) foot during two steps. *Bar Chart* classification of the primary standing foot (*light color* indicates a ground contact event). *Lower plots* pelvis lateral (body Y-direction) velocity estimates for (*1*) raw leg odometry, (*2*) filter output, (*3*) VICON-based ground truth and (*4*) error of (*1*) and (*2*) relative to VICON. *Vertical axis* units are m/s. Note that VICON velocity ground truth is of variable accuracy

independent of the walking controller, as we increased the speed of locomotion we needed to feedback the active contact points from the controller to the state estimator in this situation.

We also classify other events in the gait cycle. Striking contact is determined when a force of 20–30 N is maintained for more than 5 ms. Breaking contact is determined when force falls below 275 N. Because these impacts create unrealistic measurements, the EKF integrates these measurements with higher measurement covariance. We note that when the robot is in a double support stance, information from both legs could be leveraged to provide additional kinematic measurements. For simplicity we currently neglect this information.

Figure 7 contains a number of plots comparing (1) the raw pelvis velocity measurements inferred from kinematics with (2) the output of our integrating filter and (3) the velocity estimated from VICON motion capture. Using only raw leg position signals, the typical pelvis velocity standard deviation measured while standing stationary was 7.6 cm/s.

Adding joint-level filters reduced this to 2.3 cm/s. Integrating filtered leg joint position into the EKF further reduced the error standard deviation to 1.4 cm/s.

Finally, as mentioned in Sect. 5.5 the revolute joints states are filtered separately. The compliance model mentioned above is again used in preprocessing, before extended Kalman Filtering.

5.5.2 LIDAR

While the drift of the combined inertial and kinematic estimator is capable of achieving relatively low drift, it remains unsuitable for accurate walking over tens of meters. We aim to use our exteroceptive sensors to remain localized with the robot's environment. In particular, we use LIDAR to continuously infer the robot's position relative to a prior map while walking.

We cannot assume that the sensor is oriented horizontally (Dellaert et al. 1999), nor can we afford time to stop moving and perform static 3D registration, e.g., using an Iterative Closest Point algorithm (Besl and McKay 1992). Instead we aim to incorporate information from each individual LIDAR scan into the state estimate using a Gaussian Particle Filter (GPF), as originally proposed by Bry et al. (2012).

In typical operation, the robot is first commanded to stand still for about 30 s while it collects a full 3D point cloud of its environment (see Fig. 8). This cloud is then converted into a probabilistic occupancy grid (OctoMap) (Wurm et al. 2010) against which efficient localization comparisons later performed during locomotion. While the MAV experiments presented in Bry et al. (2012) required offline mapping with a separate sensor, our legged humanoid and actuated LIDAR with 30 m range permit the map to be constructed immediately prior to operation and immediately utilized while walking. Furthermore, if the robot were to approach the map
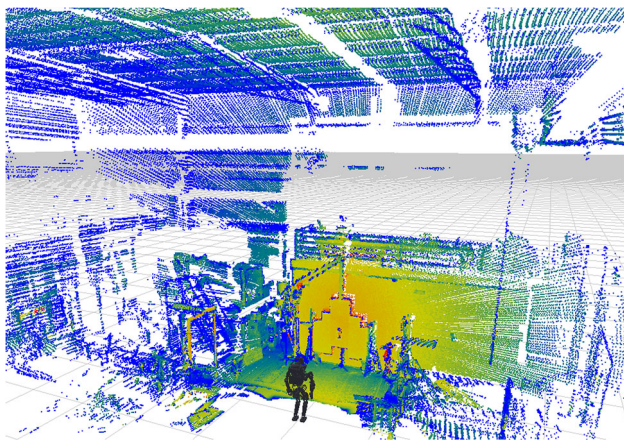
boundary, on-line construction of a new map could easily be performed during operation.

Since the LIDAR is fundamentally a planar 2D sensor, only a subset of the state vector (namely x, y and yaw in the rotating sensor plane) is observable at any given instant. We therefore partition the full state vector into observable and unobservable sub-states, and use a GPF to incorporate each laser measurement over the observable variables. The particle filter samples are weighted according to the proposed sub-state likelihood, which is computed by comparing the LIDAR measurements, projected from the sub-state, to the prior map. From these weighted samples a mean and covariance, and in turn an equivalent Kalman measurement update for the full state vector are calculated resulting in a correction to the base position and yaw.

One technical note is that our projection of LIDAR range returns as points in the 3D workspace accounts for the robot's motion, and more importantly, the spindle rotation during the 1/40 s scanning period of the internal mirror of the sensor. Neglecting this effect would result in mis-projections of returns to the side of the robot by as much at 2.5 m at the highest spindle rotation speed. Accurate projection also requires precise calibration of the LIDAR sensor, as discussed in Fallon et al. (2015).

Given the competition environment of the DARPA Robotics Challenge and our inability to test in precisely the DRC conditions (outdoors with a large crowd of moving people), we typically operated without the LIDAR localization module.

Without this information, the robot's position is subject to drift. While linear position drift rate was satisfactorily low, the yaw of the robot is unconstrained by any kinematic measurements and is subject to a slow persistent rotation drift. In particular while standing still for tens of minutes during testing, a few degrees of error can accrue causing fixtured objects (e.g. terrain height-maps) in the scene to become incorrectly positioned.

5.6 Latency and computation

Using the presented state estimator in a real-time system requires careful consideration of latency. The LIDAR range measurements require significantly more time to be sensed and processed, which introduces significant latency relative to the 1 kHz kinematic and inertial information. These latencies are shown in Table 2 for a 3.3 GHz 12-core desktop PC. We use a multi-process messaging architecture to parallelize computation, with the GPF algorithm requiring a single CPU core. Within the estimator, the EKF retains a 1 s history of measurements to accommodate the LIDAR/GPF latency.

The values and the experiments presented in Sect. 6.1 use 1000 GPF samples, although reliable performance (and reduced latency) is possible with just 300 samples.

Fig. 8 The robot initially collects a static LIDAR point cloud of its environment, which is then converted into an occupancy map for subsequent localization

Table 2 Latencies and frequencies of various state estimator components

Component	Latency	Frequency
Lower joint Kalman filters	0.16 ms	1 kHz
Pose extended Kalman filter	0.54 ms	333 Hz
LIDAR data transmission	7 ms	40 Hz
GPF processing time	11.4 ms	40 Hz
Overall LIDAR latency	18.4 ms	40 Hz

6 Experiments

We describe several experiments performed on the robot and in simulation. Code for our planning and control algorithms, along with a variety of simulation examples, is available for download in the Drake (2014a) toolbox.

6.1 State estimation evaluation

To characterize the state estimator we evaluate its performance in a variety of experiments. In our first set of experiments, we compare our state estimator to the estimator provided by Boston Dynamics (BDI) in a variety of walking scenarios. Because the robot's BDI estimator requires infor-

mation from their walking controller, we were unable to use our walking controller in these tests. In the following our localization estimate system sequentially updated a set of desired footstep locations that were individually passed to the BDI controller during locomotion. Figure 9 summarizes the results of a substantial set of experiments for a variety of walking patterns totaling 57 min of operation and 155 m traveled.

The kinematic-only estimates continuously drift, typically at 1.2–1.5 cm per step. This drift rate generally increases when the walking dynamically or on non-flat terrain. Orientation estimation performance is comparable between different estimators. Note that the precision of the ground truth orientation determined using VICON measurements is on the order of 1°, so we were unable to differentiate yaw drift on a finer scale than this.

As the reader can see in Fig. 9 the value of fusing LIDAR-based corrections becomes evident after just a few steps. Walking for just 10 mins, the kinematic-only estimators drift by as much as a meter while the LIDAR aided approach remains accurate to within 2 cm throughout. In the manipulation experiment, the LIDAR contribution actually degrades performance slightly due to occlusions caused by arm motions. For this reason, we typically discard LIDAR data when standing still.
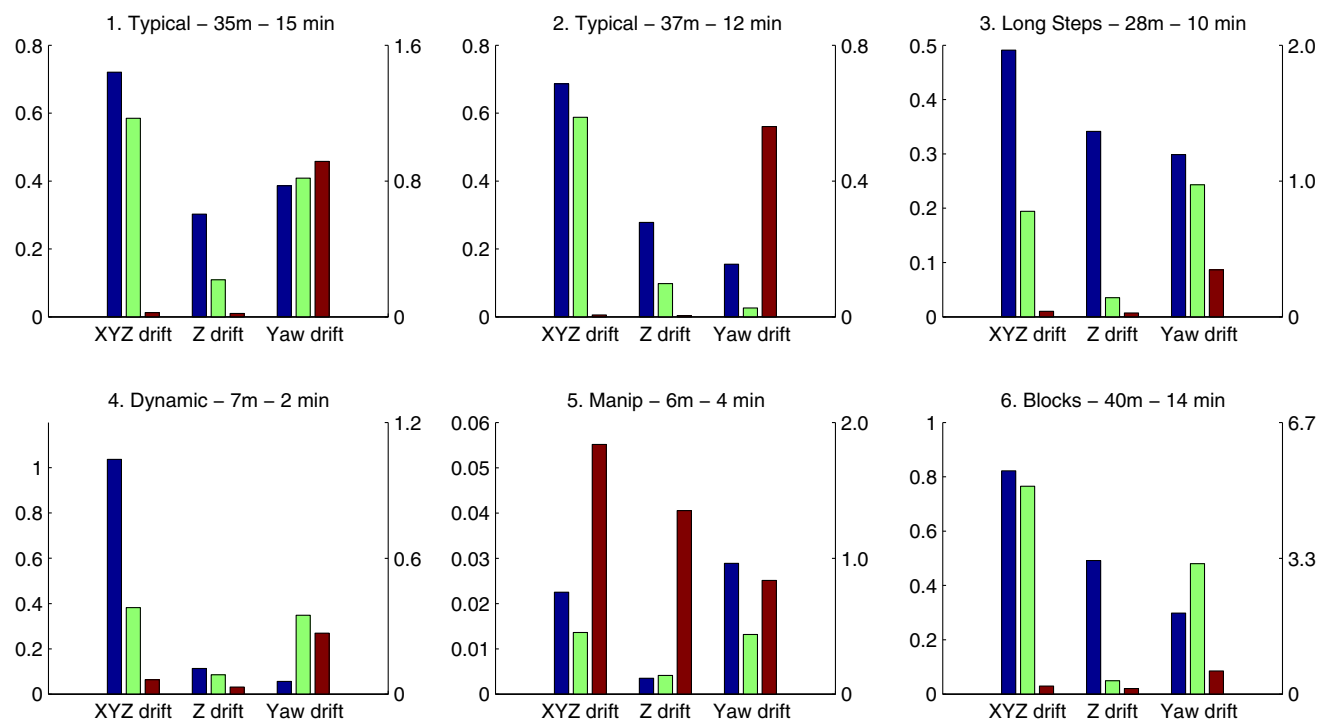

Fig. 9 Summary of localization accuracy for a variety of walking experiments. Position drift is measured with the left scale (in meters) and yaw drift with the right scale (in degrees). Error (versus ground truth) of the BDI state estimator (*blue*), our kinematic-only EKF (*green*), and

LIDAR (*red*) estimators are shown. *Clockwise* from *upper left*: *1* typical gait (15 cm forward steps), *2* typical gait with a partial map, *3* long steps (36 cm forward steps), *4* dynamic walking, *5* carrying out manipulation, *6* traversing the cinder block course

6.2 Walking and balancing control

We have evaluated the walking controller in several operating scenarios relevant to the DRC. During typical walking, we observe low-error COP and footstep tracking performance. Figure 10 illustrates the COP measured using the foot force/torque sensors during a typical walking trajectory. Note that unlike many COP-based walking controllers in the literature, we are not using the COP measurement from the foot force/torque sensor in our feedback loop. This tracking performance suggests that a model-based controller and state estimator with sufficiently low error be used to achieve stable walking with minimal force information at the foot.

We are currently capable of walking approximately 13 times faster (0.40 vs. 0.03 m/s) with 30 cm longer steps (70 cm) compared with the quasi-static walking controller provided with the robot while maintaining equal or better footstep placement precision. At this speed the double-support time per step is roughly 0.1 s and the gait is fully dynamic (the COM is outside of the support polygon). Although this is still a slow gait by human standards, we point out that our primary development goal to this point has been to achieve accurate and robust walking control in environments related to the DRC (cinder blocks, ramps, gaps, etc.) and we have not yet invested considerable effort improving the speed and efficiency of the robot's walking motions.

For traversing ramps and cinder blocks, raising the heel of the trailing foot while walking improves kinematic reachability. The controller actively monitors the knee angle and breaks heel contact when the trailing leg is nearly fully extended, resulting in a heel-raising behavior. In our implementation, the controller and state estimator are separate processes that independently estimate foot contacts (the controller uses both sensed and desired contact information to classify a foot as in contact). We therefore had to tune the contact classifier in the estimator to handle the slightly different contact behaviors encountered in flat ground, steps, and ramps.

Figure 11 highlights some test cases we have used to evaluate the robot's balancing and walking capabilities including carrying unmodeled loads, traversing irregular terrain, and climbing narrow stairs with partial foot contacts (realized by shrinking the effective foot contact surface in both the planner and controller).

6.3 Closed-loop walking with LIDAR feedback

During the December 2013 DRC Trials, most teams executed the terrain course two steps at a time. In our case at least, state estimation drift was the primary factor that motivated this conservative approach (see our previous paper for an analysis of this task (Fallon et al. 2015)).

We designed a simple experiment to compare the performance of our LIDAR-based state estimator and the kinematic-only estimator. In this experiment, the robot used each estimator in a closed loop while attempting to walk continuously up 6 cinder block steps using a fixed footstep plan. This scenario requires great precision, if the state estimator drifts by even a few centimeters, the robot will hit a step edge and fall.

Directly feeding the higher-variance low-rate LIDAR corrections through to the state estimator can result in significant jumps in the position (1–2 cm) and yaw of the robot. Instead, our approach was to maintain a second estimate exposed to the LIDAR measurements and to corrected drift in our primary estimate by slowing correcting it to stay aligned to the LIDAR-based estimate using a high covariance position correction.

Using this approach Fig. 12 illustrates the robot successfully climbing to the top of the stack of blocks. The kinematic-only estimator only reached the fourth step. At each step the state drifted characteristically backwards, resulting in the robot drifting forward. The robot's trailing foot eventually collided with the front of the step resulting in a fall.
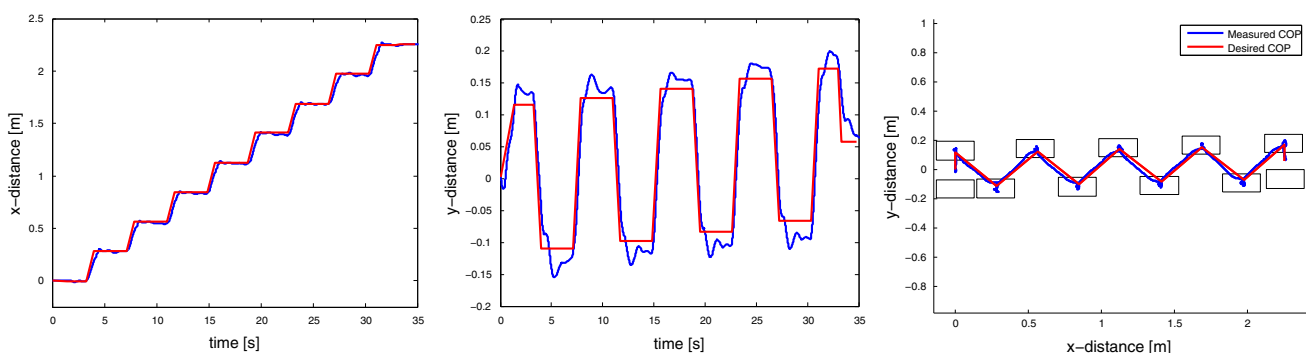

Fig. 10 Typical COP tracking performance for Atlas walking on flat terrain

Fig. 11 Example walking and balancing test cases: **a** balancing with one foot on a moving pallet jack, **b** walking while carrying an unmodeled 8.5 kg aluminum truss, **c** climbing stairs using partial foot contact, **d** traversing a set of tilted cinder blocks
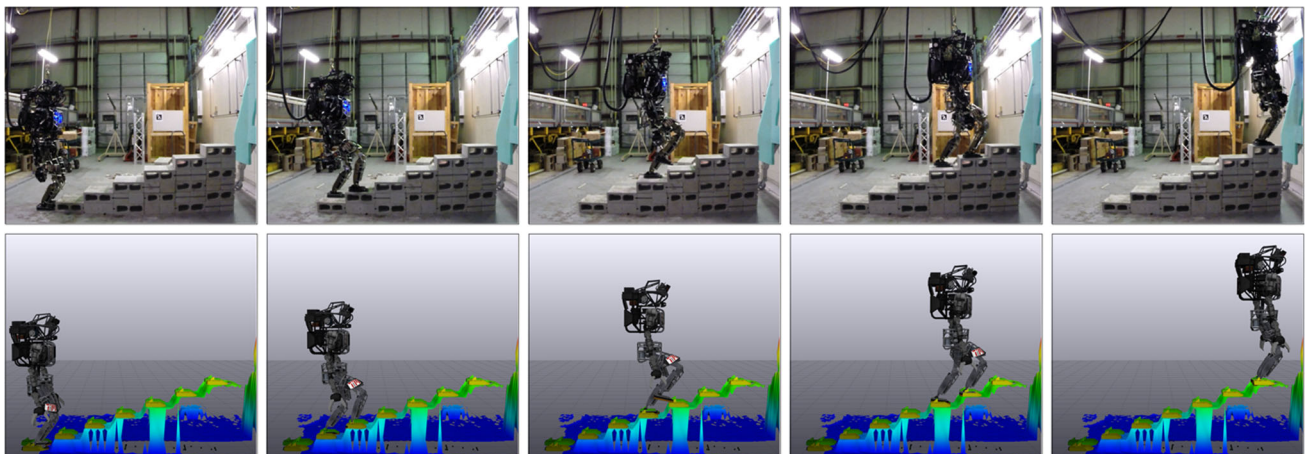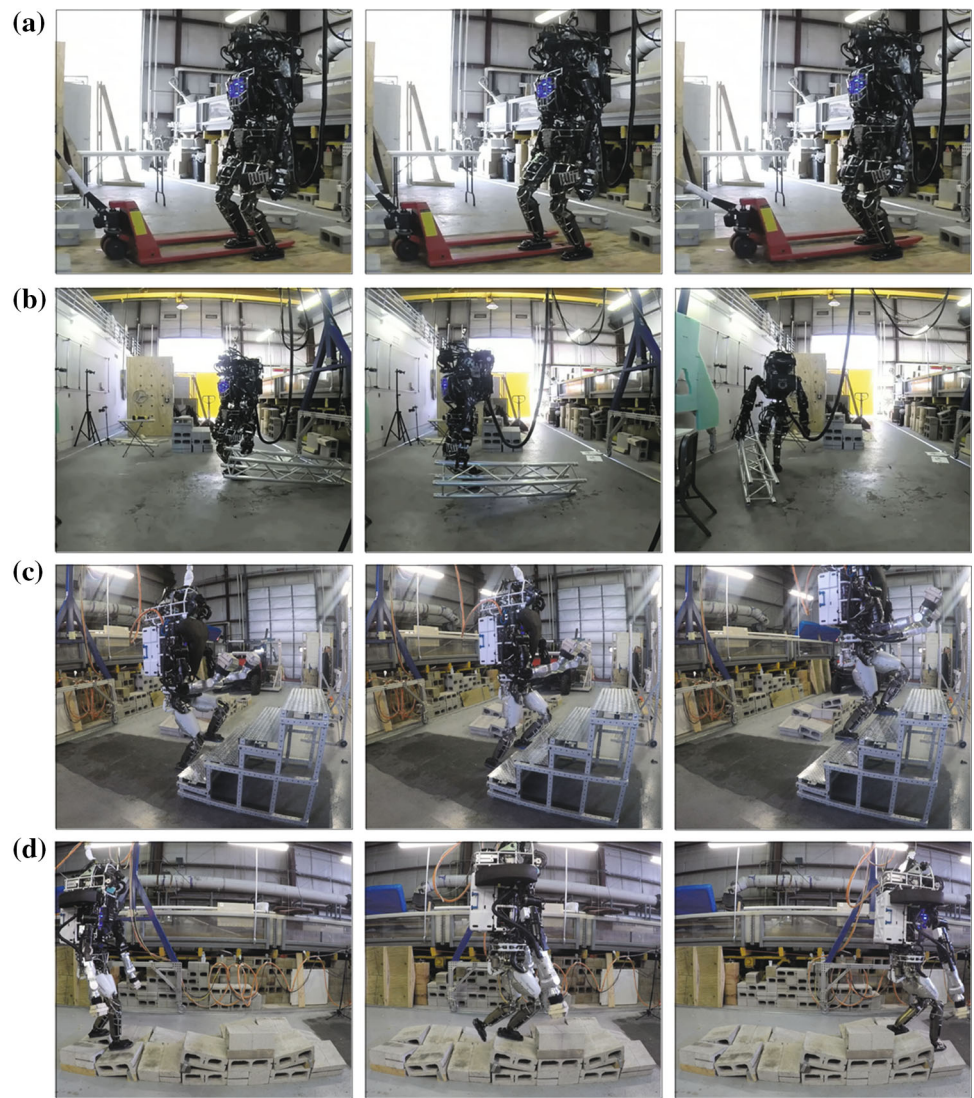

Fig. 12 Atlas walking continuously up six cinder block steps using LIDAR-based state estimation in a closed loop with the walking controller. *Top* images of the robot climbing the stack of cinder blocks in our laboratory. *Bottom* the state estimate rendering in our user interface

Fig. 13 Snapshots from a half-stride of simulated running at 2 m/s. **a** Touch-down, **b** mid-stance, **c** toe-off, **d** mid-flight
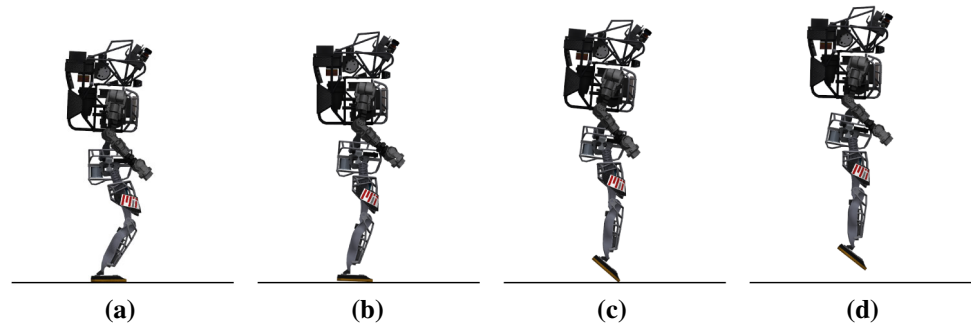

(a) (b) (c) (d)
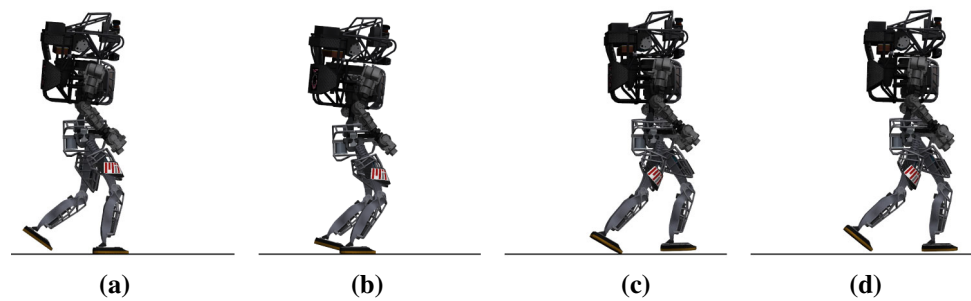
Fig. 14 Snapshots from a simulated jump. **a** Starting posture, **b** Heel-off, **c** toe-off, **d** apex

(a) (b) (c) (d)

6.4 Dynamic motion planning

We are actively developing new dynamic behaviors for the robot using the whole-body motion planning and control algorithms described previously. Here we present two results executing running and jumping motions in simulation.

6.4.1 Running

To plan a running motion for Atlas, we consider a single half stride starting at the apex of a flight phase. We fix the contact sequence to be flight, left-stance, left-toe-stance, flight. By constraining the initial and final conditions of the trajectory such that all quantities are mirrored about the robot's sagittal plane, we obtain a trajectory that can be mirrored to yield a full stride. In addition to enforcing the contact sequence and the periodicity of the trajectory, we also specify a stride-length and average speed (1.5 m and 2 m/s respectively for the gait shown in Fig. 13), require at least 3 cm of clearance between links to avoid self-collisions, and constrain the gaze of the robot's head cameras to be no more than 15° from the direction of travel. Solving for the half-stride motion takes approximately 2.5 min on a computer with a 3.3 GHz Intel i7 CPU. The controller described in Sect. 4 is used to stabilize the resulting trajectory (mirrored and looped to produce a ten-stride sequence) on a simulated model of Atlas with accurate torque limits. One half-stride of the simulated motion is depicted in Fig. 13.

6.4.2 Jumping

Figure 14 shows the first half of a jumping motion generated by the whole-body planner and stabilized by the controller from Sect. 4. This motion was generated by constraining the corners of the feet to be at least 5 cm above the ground at the mid-point of the trajectory, and requiring that the robot's motion be symmetric about its sagittal plane. Solving for the jumping motion takes approximately 1.5 min on a computer with a 3.1 GHz Intel i7 CPU. As in the running case, this entire motion was stabilized in a simulation with torque limits.

Our final example is another jumping motion, this time from a cinder block to the ground. The robot's starting posture on top of the cinder block is specified, but its final posture is not. We require that the feet maintain at 3 cm of clearance from the cinder block at all times after take-off and that they be on the ground at the end of the trajectory. Given these constraints, and the requirement that the robot's motion be symmetric about its sagittal plane, the planner finds the motion depicted in Fig. 15. This sort of maneuver will be useful in competition, as stepping down from obstacles is challenging given Atlas' leg kinematics. Solving this problem takes approximately 10 min on a computer with a 3.1 GHz Intel i7 CPU. The jump down motion was stabilized by our controller in simulation with torque limits.
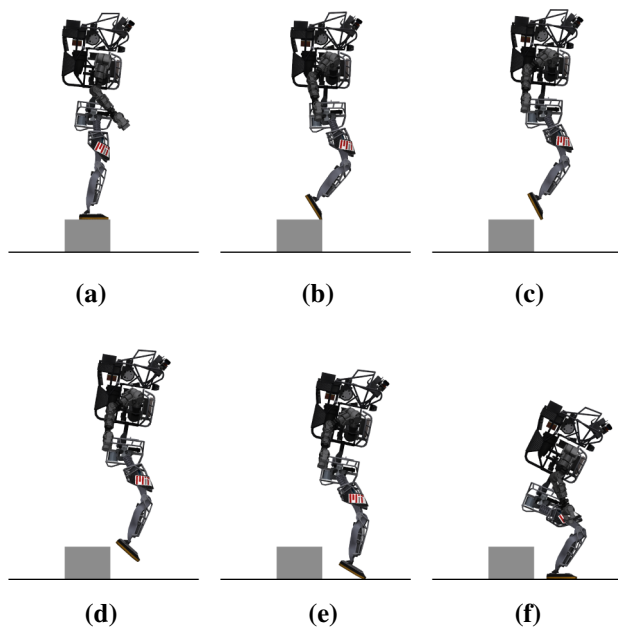
Fig. 15 Snapshots from a motion plan for jumping off a cinder block. **a** Starting posture, **b** toe-off, **c** apex, **d** avoiding collision, **e** touch-down, **f** final posture

7 Future work

Our ambitions for future work can be summarized in a few words: robustness, speed, and versatility. In terms of robustness, we have only recently begun to explore reactiveness of the robot to large external perturbations. This is a critically important feature for any dynamically-stable field robot (especially those that weigh more than 150 kg). Our approach to efficiently (re-)computing LQR solutions online combines nicely with standard approaches to step recovery on flat terrain. However, recovering safely in cluttered environments and on irregular terrain remains an exciting open question.

Another potential improvement has to do with the method by which we integrate LIDAR corrections. Currently, we input low-pass filtered corrections from the LIDAR into the estimated state used by the controller. Alternatively, one could use receding horizon footstep planning to adjust the plan based on LIDAR corrections without directly modifying the state estimate that is input to the controller. We expect this approach would be significantly more robust in the presence of rapid state drift.

We are excited about improving the locomotion speed of Atlas to match what we have been able to achieve in simulation (over 1.0 m/s). Our hypothesis is that improved system identification (so as to reduce our dependence on the integrated joint velocity references) and better leg swing and COP trajectories will contribute significantly to closing this gap.

We also intend to translate the running and jumping behaviors to hardware and explore dynamic multi-contact motions, such recovering after a fall. For the latter goal, we expect to generalize our state estimator to reason about contacts distributed throughout the robot's body. Stabilizing whole-body plans in hardware will demand general techniques for achieving torque-feasibility without significant human interaction, possibly by processing the output of our dynamic motion planner. This is an important open problem that still must be addressed.

8 Conclusion

In this paper we described the optimization algorithms that comprise our approach to achieving reliable locomotion in demanding environments with varied terrain and geometric constraints. For planning typical walking motions, our footstep planner efficiently solves a mixed-integer convex optimization to compute an obstacle-free sequence of footsteps from estimates of the terrain. For more complex kinematically-constrained, multi-contact motions, we proposed a sparse nonlinear trajectory optimization algorithm that combines full body kinematics with centroidal dynamics to efficiently compute whole-body dynamic motions.

Our controller exploits time-varying LQR solutions on reduced dynamical models to construct an efficiently-solvable QP to stabilize both walking and whole-body trajectories. Despite significant sensor limitations and inevitable model inaccuracies, we have been able to achieve reliable balancing and walking with Atlas. Notably, by combining IMU, kinematic, and LIDAR data within our state estimator, we have demonstrated precise navigation over extended walking trajectories. Our current efforts are focused on improving both the speed and variety of dynamic locomotion strategies.

References

Abe, Y., da Silva, M., & Popović, J. (2007). Multiobjective control with frictional contacts. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 249–258). Switzerland: Aire-la-Ville.

Ames, A. D. (2012). First steps toward underactuated human-inspired bipedal robotic walking. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)*. St. Paul, MN.

Ames, A. D. (2013). Human-inspired control of bipedal robotics via control Lyapunov functions and quadratic programs. In *Hybrid systems: Computation and control*.

Ames, A. D., Galloway, K., & Grizzle, J. W. (2012). Control Lyapunov functions and hybrid zero dynamics. In *Proceedings of the 51st IEEE conference on decision and control*. HI: Maui.

Bartlett, R. A., Wächter, A., & Biegler, L. T. (2000). Active set vs. interior point strategies for model predictive control. In *Proceedings of the American control conference*. Chicago, IL.

Baudouin, L., Perrin, N., Moulard, T., Lamiraux, F., Stasse, O., & Yoshida, E. (2011). Real-time replanning using 3d environment for humanoid robot. In *IEEE-RAS international conference on humanoid robots* (pp. 584–589). Bled, Slovnie.

Bemporad, A., & Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, *35*(3), 407–427. doi:10.1016/S0005-1098(98)00178-2.

Bertsekas, D. P. (1995). *Dynamic programming and optimal control*. Belmont, MA: Athena Scientific.

Besl, P. J., & McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 239–256.

Betts, J. T. (1998). Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, *21*(2), 193–207.

Bloesch, M., Hutter, M., Hoepflinger, M. A., Leutenegger, S., Gehring, C., Remy, C. D., & Siegwart, R. (2012). State estimation for legged robots—consistent fusion of leg kinematics and IMU. In *Robotics: Science and systems (RSS)*

Bortz, J. (1971). A new mathematical formulation for strapdown inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems*, *AES–7*(1), 61–66.

Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Bry, A., Bachrach, A., & Roy, N. (2012). State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *IEEE International conference on robotics and automation (ICRA)* (pp. 1–8).

Chestnutt, J., Kuffner, J., Nishiwaki, K., & Kagami, S. (2003). Planning biped navigation strategies in complex environments. In *Proceedings of IEEE international conference on humanoid robots, Karlsruhe, Germany*.

Chestnutt, J. E., Nishiwaki, K., Kuffner, J., & Kagami, S. (2007). An adaptive action model for legged navigation planning. In *IEEE-RAS international conference on humanoid robots* (pp. 196–202).

Collette, C., Micaelli, A., Andriot, C., & Lemerle, P. (2007) Dynamic balance control of humanoids for multiple grasps and non coplanar frictional contacts. In *Proceedings of the IEEE/RAS international conference on humanoid robots* (pp. 81–88).

Dai, H., Valenzuela, A., & Tedrake, R. (2014). Whole-body motion planning with centroidal dynamics and full kinematics. In *Proceedings of the IEEE-RAS international conference on humanoid robots*.

Dalibard, S., Khoury, A. E., Lamiraux, F., Nakhaei, A., Taïx, M., & Laumond, J. P. (2013). Dynamic walking and whole-body motion planning for humanoid robots: An integrated approach. *International Journal of Robotics Research*, *32*(9–10), 1089–1103.

Deits, R., & Tedrake, R. (2015). Efficient mixed-integer planning for UAVs in cluttered environments. In *IEEE international conference on robotics and automation (ICRA)*.

Deits, R. L. H., & Tedrake, R. (2014). Computing large convex regions of obstacle-free space through semidefinite programming. In *Proceedings of the eleventh international workshop on the Algorithmic Foundations of Robotics, Istanbul*.

Deits, R. L. H., & Tedrake, R. (2014). Footstep planning on uneven terrain with mixed-integer convex optimization. In *Proceedings of the IEEE-RAS international conference on humanoid robots*.

Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo localization for mobile robots. In *IEEE international conference on robotics and automation (ICRA)*.

Fallon, M., Kuindersma, S., Karumanchi, S., Antone, M., Schneider, T., Dai, H., et al. (2015). An architecture for online affordance-based perception and whole-body planning. *Journal of Field Robotics*, *32*, 229–254.

Fallon, M. F., Antone, M., Roy, N., & Teller, S. (2014). Drift-free humanoid state estimation fusing kinematic, inertial and LIDAR sensing. In *Proceedings of IEEE international conference on humanoid robots*

Feng, S., Xinjilefu, X., Huang, W., & Atkeson, C. G. (2013). 3D walking based on online optimization. In *Proceedings of the IEEE-RAS international conference on humanoid robots, Atlanta, GA*.

Ferreau, H., Bock, H., & Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, *18*(8), 816–830.

Garimort, J., & Hornung, A. (2011). Humanoid navigation with dynamic footstep plans. In *2011 IEEE international conference on robotics and automation (ICRA)* (pp. 3982–3987). IEEE.

Gill, P. E., Murray, W., & Saunders, M. A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, *47*(1), 99–131.

Gurobi Optimization, Inc. (2014). *Gurobi optimizer reference manual*. Retrieved, from http://www.gurobi.com/.

Herdt, A., Diedam, H., Wieber, P. B., Dimitrov, D., Mombaur, K., & Diehl, M. (2010). Online walking motion generation with automatic foot step placement. *Advanced Robotics*, *24*(5–6), 719–737.

Herzog, A., Righetti, L., Grimminger, F., Pastor, P., & Schaal, S. (2013). Momentum-based balance control for torque-controlled humanoids. *CoRR, abs/1305.2042*.

Herzog, A., Righetti, L., Grimminger, F., Pastor, P., & Schaal, S. (2014). Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In *Proceeedings of 2014 IEEE/RSJ international conference on intelligent robots and systems*.

Hornung, A., Wurm, K. M., & Bennewitz, M. (2010). Humanoid robot localization in complex indoor environments. In *IEEE/RSJ international conference on intelligent robots and systems (IROS), Taipei, Taiwan*.

IBM Corp. (2010). *User's manual for CPLEX*. Retrieved, from http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r2/topic/com.ibm.common.doc/doc/banner.htm.

Johnson, M., Shrewsbury, B., Bertrand, S., Wu, T., Duran, D., Floyd, M., et al. (2015). Team IHMC's lessons learned from the DARPA Robotics Challenge trials. *Journal of Field Robotics*, *32*, 192–208.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., & Hirukawa, H. (2003). Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In *Proceedings of intelligent robots and systems (IROS)*.

Koolen, T., Smith, J., Thomas, G., Bertrand, S., Carff, J., Mertins, N., et al. (2013). Summary of team ihmc s virtual robotics challenge entry. In *Proceedings of the IEEE-RAS international conference on humanoid robots*. Atlanta, GA: IEEE.

Koolen, T., Smith, J., Thomas, G., Bertrand, S., Carff, J., Mertins, N., et al. (2013). Summary of team IHMC's virtual robotics challenge entry. In *Proceedings of the IEEE-RAS international conference on humanoid robots, Atlanta, GA*.

Koyanagi, K., Hirukawa, H., Hattori, S., Morisawa, M., Nakaoka, S., Harada, K., & Kajita, S. (2008). A pattern generator of humanoid robots walking on a rough terrain using a handrail. In *IEEE/RSJ international conference on intelligent robots and systems, 2008, IROS 08* (pp. 2617–2622).

Kudoh, S., Komura, T., & Ikeuchi, K. (2002). The dynamic postural adjustment with the quadratic programming method. In *International conference on intelligent robots and systems (IROS)* (pp. 2563–2568).

Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., & Inoue, H. (2001). Footstep planning among obstacles for biped robots. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (vol. 1, pp. 500–505). Maui, Hawaii.

Kuffner, J. J., Nishiwaki, K., Kagami, S., Inaba, M., & Inoue, H. (2003). Online footstep planning for humanoid robots. *IEEE International Conference on Robotics and Automation*, *1*, 932–937. doi:10.1109/ROBOT.2003.1241712.

Kuindersma, S., Permenter, F., & Tedrake, R. (2014). An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *Proceedings of the international conference on robotics and automation (ICRA), Hong Kong, China*.

Lee, S. H., & Goswami, A. (2012). A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Autonomous Robots*, *33*, 339–414.

Lien, J. M., & Amato, N. M. (2004). Approximate convex decomposition of polygons. *Proceedings of the twentieth annual symposium on computational geometry* (pp. 17–26).

Lingas, A. (1982). The power of non-rectilinear holes. In M. Nielsen & E. M. Schmidt (Eds.), *Automata, languages and programming*. Lecture Notes in Computer Science (Vol. 140, pp. 369–383). Berlin Heidelberg: Springer. Retrieved, from http://link.springer.com/chapter/10.1007/BFb0012784.

Macchietto, A., Zordan, V., & Shelton, C. R. (2009). Momentum control for balance. In *Transactions on Graphics/ACM SIGGRAPH*.

Manchester, I. R., Tobenkin, M. M., Levashov, M., & Tedrake, R. (2011). Regions of attraction for hybrid limit cycles of walking robots. In *Proceedings of the 18th IFAC world congress*.

Michel, P., Chestnutt, J., Kuffner, J., & Kanade, T. (2005). Vision-guided humanoid footstep planning for dynamic environments. *IEEE-RAS international conference on humanoid robots* (pp. 13–18).

Mombaur, K. D. (2009). Using optimization to create self-stable human-like running. *Robotica*, *27*(3), 321–330.

Mosek ApS. (2014). *The MOSEK optimization software*. Retrieved, from http://www.mosek.com/

Neo, E., Yokoi, K., Kajita, S., & Tanie, K. (2007). Whole-body motion generation integrating operator's intention and robot's autonomy in controlling humanoid robots. *IEEE Transactions on Robotics*, *23*(4), 763–775. doi:10.1109/TRO.2007.903818.

Nishiwaki, K., Chestnutt, J., & Kagami, S. (2011). Autonomous navigation of a humanoid robot on unknown rough terrain. In *Proceedings of the international symposium on robotics research*.

Orin, D. E., & Goswami, A. (2008). Centroidal momentum matrix of a humanoid robot: Structure and properties. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems, Nice, France*.

Pollard, N. S., & Reitsma, P. S. A. (2001). Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model. In *Yale workshop on adaptive and learning systems*.

Posa, M., Cantu, C., & Tedrake, R. (2014). A direct method for trajectory optimization of rigid bodies through contact. *International Journal of Robotics Research*, *33*(1), 69–81.

Rotella, N., Bloesch, M., Righetti, L., & Schaal, S. (2014). State estimation for a humanoid robot. *CoRR, 1402.5450*. IROS Submission.

Saab, L., Ramos, O., Mansard, N., Fourquet, J. Y., & Souères, P. (2011). Generic dynamic motion generation with multiple unilateral constraints. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems, San Fransisco, USA*.

Saab, L., Ramos, O. E., Keith, F., Mansard, N., Souères, P., & Fourquet, J. Y. (2013). Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Transactions on Robotics*, *29*(2), 346–362.

Shkolnik, A., Levashov, M., Manchester, I. R., & Tedrake, R. (2011). Bounding on rough terrain with the littledog robot. *The International Journal of Robotics Research (IJRR)*, *30*(2), 192–215.

Stasse, O., Davison, A. J., Sellaouti, R., & Yokoi, K. (2006). Real-time 3D SLAM for a humanoid robot considering pattern generator information. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Stephens, B. J. (2011). State estimation for force-controlled humanoid balance using simple models in the presence of modeling error. In *IEEE international conference on robotics and automation (ICRA)* (pp. 3994–3999).

Tedrake, R. (2014). *Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems*. Retrieved, from http://drake.mit.edu.

Tedrake, R. (2014). *Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832)*. Retrieved in Fall, 2014 from http://people.csail.mit.edu/russt/underactuated/.

Wang, Y., & Boyd, S. (2011). Fast evaluation of quadratic control-Lyapunov policy. *IEEE Transactions on Control Systems Technology*, *19*(4), 939–946.

Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., & Burgard, W. (2010). OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proceedings of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation, Anchorage, AK, USA*.

Xinjilefu, X., Feng, S., Huang, W., & Atkeson, C. (2014). Decoupled state estimation for humanoids using full-body dynamics. In *IEEE international conference on robotics and automation (ICRA), Hong Kong, China*.

Scott Kuindersma is an Assistant Professor of Engineering and Computer Science at Harvard University. From 2012-2015 he was a postdoc in the Robot Locomotion Group at MIT CSAIL and the Planning and Control Lead for the MIT DARPA Robotics Challenge team. He received his MS and PhD in Computer Science from the University of Massachusetts Amherst. His research interests broadly encompass legged robotics, optimization, control, nonlinear systems, and machine learning.

Robin Deits is a Ph.D. student in the Computer Science and Artificial Intelligence Laboratory at MIT and a Hertz Foundation Fellow. He received an S.B. in Physics from MIT in 2011 and an M.S. in Electrical Engineering and Computer Science from MIT in 2014. As a member of the Robot Locomotion Group, he studies planning and control for walking, running, and flying robots.
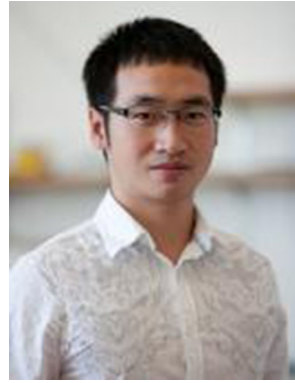
Hongkai Dai is currently a PhD student in Computer Science and Artificial Intelligence Lab, MIT. He got B.Eng from Hong Kong University of Science and Technology, and M.Sc from MIT. His current research focuses on robust motion planning.

Maurice Fallon is a Lecturer (Assistant Professor) in Robotics and Computer Vision in the School of Informatics at the University of Edinburgh. He received his Ph.D. degree in statistical signal processing in 2008 from the University of Cambridge. Between 2008 and 2014 he worked at MIT in the Computer Science and Artificial Intelligence Laboratory. His publications were nominated for best paper nominations at the International Conference on Robotics and Automation (ICRA) in 2011, 2012 and 2013. From 2013-2015 he was the perception lead of MIT's DARPA Robotics Challenge team which finished in 6th place in the final competition. His research is focused on state estimation, mapping and humanoid motion planning.

Frank Permenter is a graduate student at MIT, where he's a member of the Computer Science and Artificial Intelligence Laboratory (CSAIL) and the Laboratory for Information & Decision Systems (LIDS). He is advised by Prof. Pablo Parrilo and Prof. Russ Tedrake. In 2006, he received the Bachelor of Science degree in electrical engineering from the University of Houston, and, in 2010, a Master of Science degree in electrical engineering from Stanford. From 2004-2007, he was a member of the guidance, navigation, and control flight software team for the International Space Station, working on the attitude (orientation) control system. From 2007-2011, he helped develop the Robonaut 2 system, a robotic astronaut now resident on the International Space Station. His research interests include convex optimization, algebraic methods, and their applications to robotics.

Andrés Valenzuela is a doctoral candidate in the Department of Mechanical Engineering at the Massachusetts Institute of Technology. He received his B.S. in Mechanical Engineering from the University of Notre Dame in 2009, and his S.M. in Mechanical Engineering from MIT in 2011. He is a member of the Robot Locomotion Group in the Computer Science and Artificial Intelligence Lab at MIT, and participated in MIT's entry to the DARPA Robotics Challenge from 2013 to 2015. His research focus is on exploiting structure in dynamic motion planning for legged robots, with the goal of providing efficient methods for planning highly dynamic motions in complicated environments.

Twan Koolen is currently a Ph.D. student in the Computer Science and Artificial Intelligence Laboratory at MIT and a member of the Robot Locomotion Group. Twan received his M.Sc. degree in mechanical engineering cum laude from Delft University of Technology in 2011, and has worked as a research associate at the Institute for Human and Machine Cognition between 2010 and 2014. His main research interest is dynamic and robust legged locomotion.

Pat Marion is a Ph.D. student in the Computer Science and Artificial Intelligence Laboratory at MIT. He received a B.S. in Computer Science from Rensselaer Polytechnic Institute in 2008. Pat joined the MIT DARPA Robotics Challenge team in 2013 as a Robotics Software Engineer. As a member of the Robot Locomotion group at MIT Pat studies perception and planning.

Russ Tedrake is the X Consortium Associate Professor of Electrical Engineering and Computer Science, Aeronautics and Astronautics, and Mechanical Engineering at MIT, the Director of the Center for Robotics at the Computer Science and Artificial Intelligence Lab, and the leader of Team MIT's entry in the DARPA Robotics Challenge. He is a recipient of the NSF CAREER Award, the MIT Jerome Saltzer Award for undergraduate teaching, the DARPA Young Faculty Award in Mathematics, the 2012 Ruth and Joel Spira Teaching Award, and was named a Microsoft Research New Faculty Fellow.