Abstract on  
“Locomotion Planning, Optimization and Control Design for Humanoid Robot”

# Introduction

Atlas is a full-size hydraulic humanoid robot built by Boston Dynamics, Inc. Here is described the complete system integration on Atlas. The goal 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), Boston Dynamics were 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 described the 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. To move around the environment the humanoid robot must be able to walk and keep balance. This abstract will describe the approach to implement a two-legged robot walking system, which Boston Dynamics used to construct Atlas.

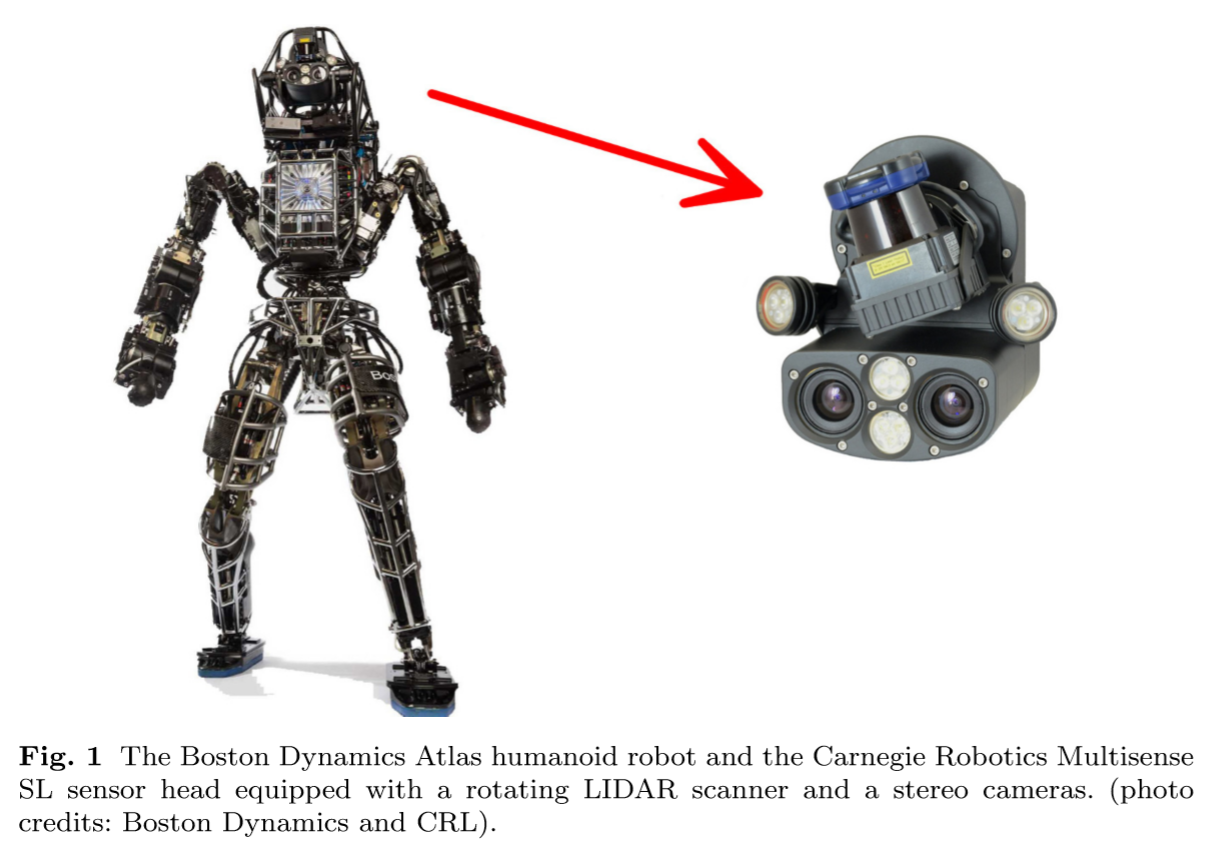
# Approach overview

The 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.

A time-varying linear quadratic regulator (LQR) design is used to stabilize trajectories for a simplified dynamic model of the robot. 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), the stabilizing properties of LQR exploited while maintaining the versatility a aﬀorded by QP-based control formulations in which whole-body motions can be tracked or constrained in a variety of ways. To implement the controller on a physical system requires that being able to efficiently compute solutions to the QP at each control step. Boston Dynamics used 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. With such approach of footstep computation, the robot is capable of walking over nontrivial terrain while maintaining extremely low drift from the desired footstep trajectory, which is a critically important capability to navigate efficiently through obstacle-ridden environments.



# Atlas physical architecture

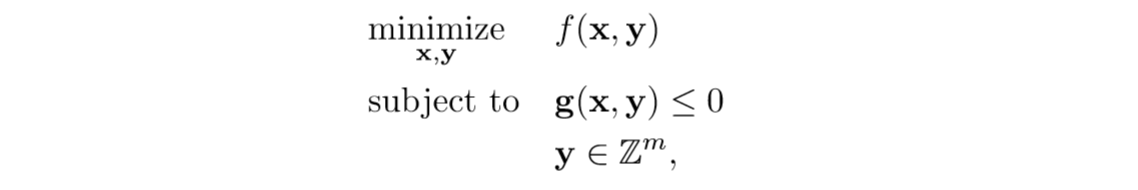
Atlas is a full-scale, hydraulically-actuated humanoid robot manufactured by Boston Dynamics, Inc. The robot stands approximately 188cm tall with a total mass of 155kg (without hands attached). It has 28 actuated degrees of freedom: 6 in each leg and arm, 3 in the back, and 1 neck joint. A tether attached 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 1000Hz 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. 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 Figure 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 30RPM. The LIDAR captures 40 scan lines of the environment per second, each containing 1081 range returns out to a maximum range of 30 meters. 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. Most 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 25kg in additional mass, and capability to operate wirelessly with battery power and onboard computers for perception and control.

# Optimized 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 which are interesting in the scope of current task. 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.

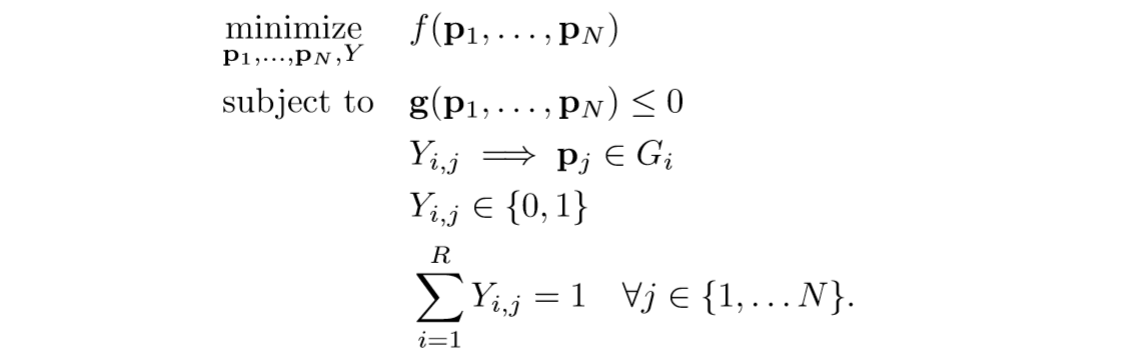
## Footstep Planning as a Mixed-Integer Convex Problem

The footstep planning problem is identified 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. 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. 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. However, the need for convex constraints prevents this optimization from considering the yaw of the robot or its feet and prevents it from handling obstacle avoidance. One of the Boston Dynamics prior works 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. Prior to planning any walking motion, the area is classified around Atlas as safe or unsafe for footstep placement. In the environments used for the DRC, was found it suﬃcient to simply exclude areas of the terrain that are steeper than a predeﬁned 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 nonconvex 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. 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. 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. The general form of mixed-integer convex programming is:



where f and g are convex functions and the elements of the vector y ∈ Zm take on integer values.

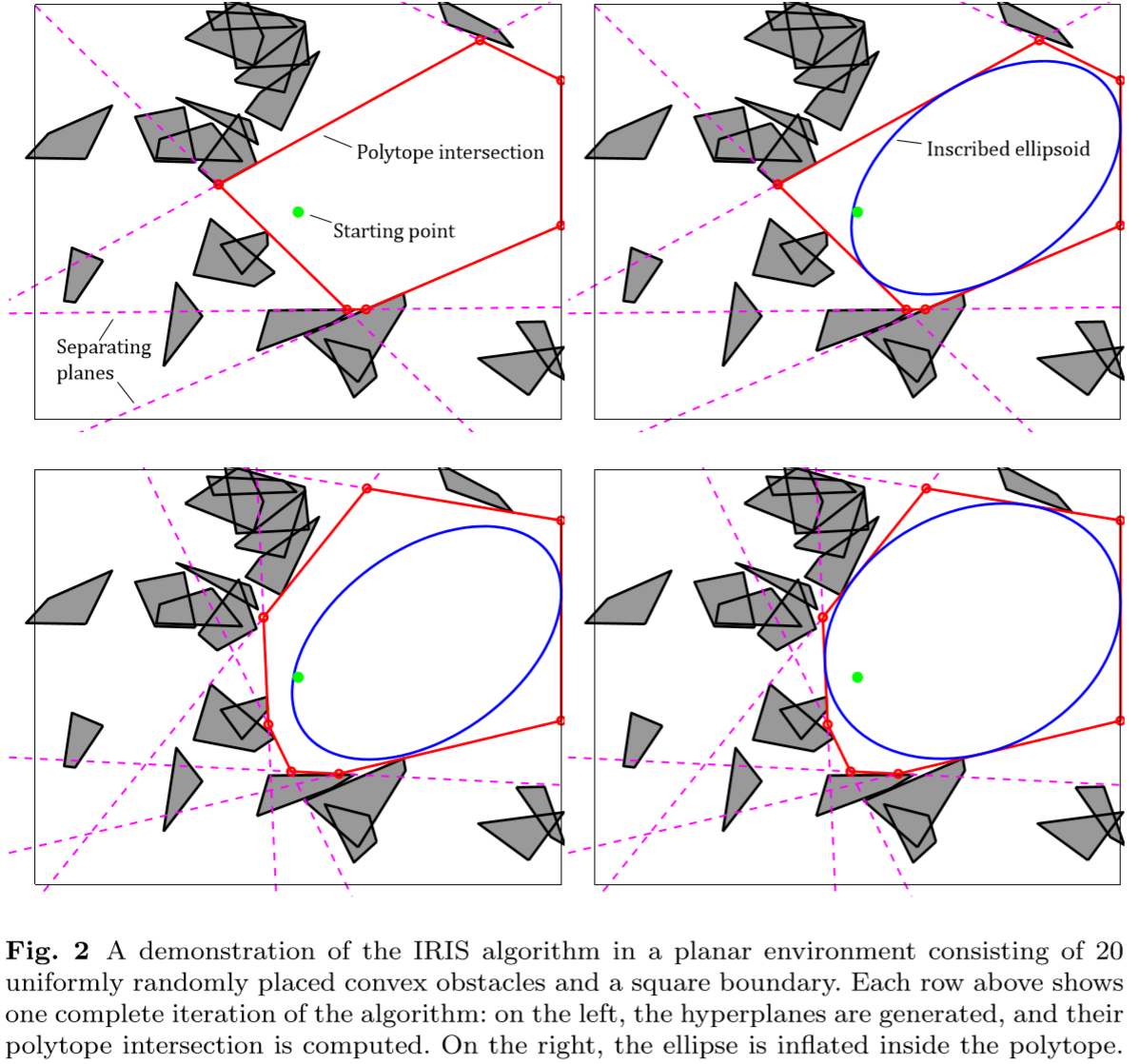
A special case of this is mixed-(0,1) convex programming in which they are restricted to values of only 0 or 1. Atlas footstep planning system uses binary variables of this form to indicate the assignment of footsteps to regions. Let p1, p2, ..., pN be the poses of the footsteps, expressed as position and yaw with pj = (xj, yj, zj, θj), and let G1, G2, ..., GR be the regions of safe terrain, represented as convex polytopes. The matrix Y∈{0,1}R×N represents the assignment of footsteps to safe regions. The optimization problem is:



The conditional constraint that Yi,j ⇒ pj ∈ Gi can be represented exactly using a standard big-M formulation, provided that we have some bounds on the possible values of the pj. 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 g (p1, ..., pN) ≤ 0 represent an approximation of the reachable set of footsteps for Atlas.

## Convex Decomposition

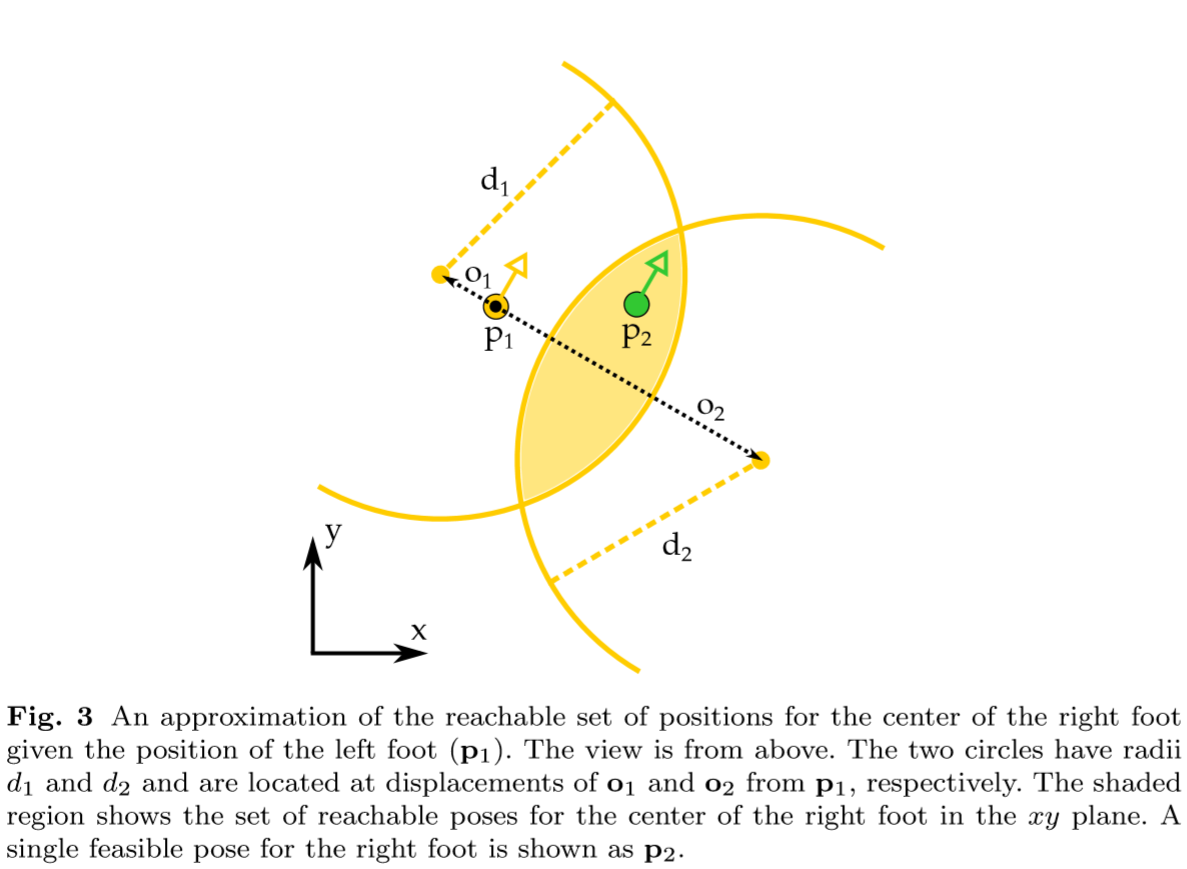
To simplify the combinatorial problem of assigning footsteps to convex safe regions, we would like to minimize the number of convex pieces into which the safe terrain set is decomposed. This presents a number of challenges. First of all, even for a two-dimensional environment with polygonal obstacles, 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. 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. 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. 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 Figure 2.



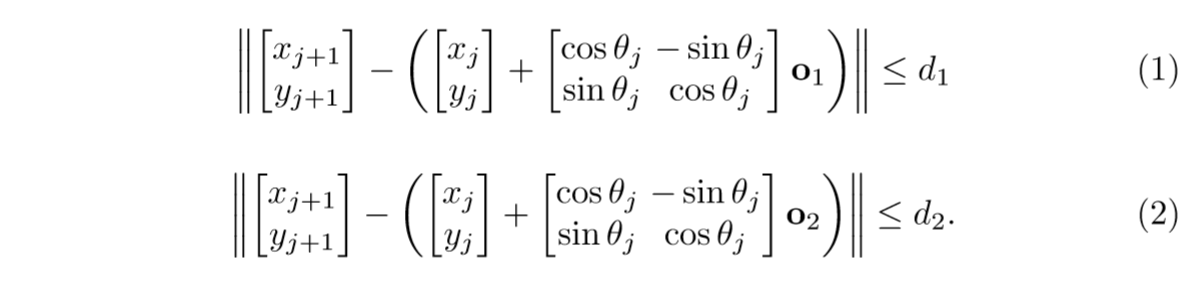
The result of IRIS is the ﬁnal polytope or ellipsoid, either of which can be used as a convex representation of obstacle-free space. We use the polytope representation in our planner, since it is always of larger volume than the (inscribed) ellipsoid and can be represented as a set of linear constraints.

## Representing Reachability

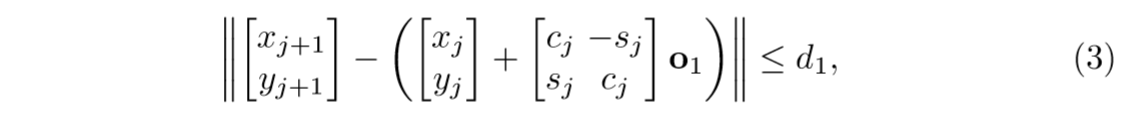
When planning footstep placements, we must somehow represent the kinematic reachability of the robot, that is, the set of foot placements that can be achieved given the constraints imposed by the dimensions of the limbs and the limits of the joints. Directly reasoning about this reachable set using the 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 dk and is located at some ﬁxed oﬀset ok in the frame of the prior footstep. The reachable region deﬁned by these circles is shown in Figure 3. For each footstep j, we require that



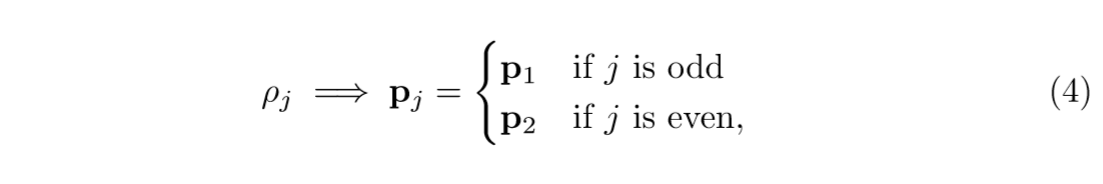
This is not yet a convex constraint, since cos and sin are non-convex functions. To mitigate this, we replace sin(θj) and cos(θj) with additional decision variables, which we label sj and cj, respectively. Equation 1 becomes



which is a convex quadratic constraint. Of course, we still must ensure that sj and cj 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 θj. By choosing the number of piecewise linear segments that we use in our approximation, we can trade oﬀ between accuracy and computational speed.

## Determining the Number of Footsteps

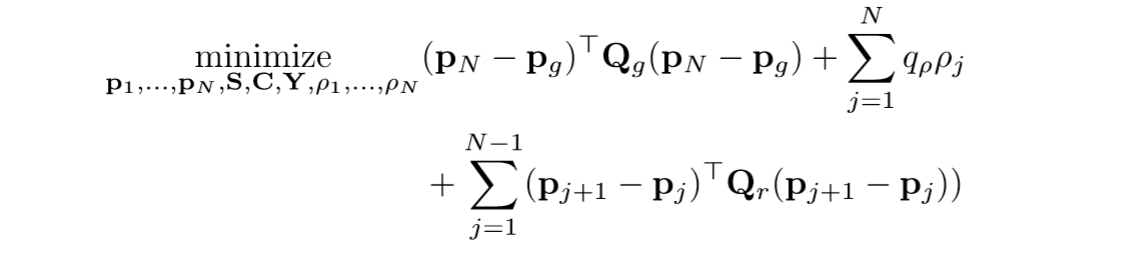
In general, we cannot expect to know a priori how many footsteps will be needed to reach a target position, so the footstep planner must be responsible for determining this number. Since the entire set of footsteps is simultaneously optimized, changing the number of footsteps alters the size of the optimization 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 step is unused. We label this ﬂag ρj and require that if ρj is true, then footstep j be ﬁxed to the starting pose of that foot



where p1 and p2 are the ﬁxed initial poses of the feet. Adding a negative cost on each ρj 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 ρj equal to 1 can be removed from the plan.

## Final Footstep Planning Optimization Task

We formulate the entire footstep planning optimization as follows:



where pg ∈ R4 is the x, y, z, θ goal pose, Qg ∈ S4 and Qr ∈ S4 are objective weights on the distance to the goal and between steps, qt ∈ R is an objective weight on trimming unused steps and the full expression is subject to, for j = 1, ..., N:

* safe terrain regions:



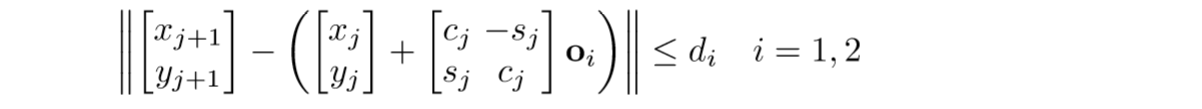
* piecewise linear sin(θ):



* piecewise linear cos(θ):



* approximate reachability:

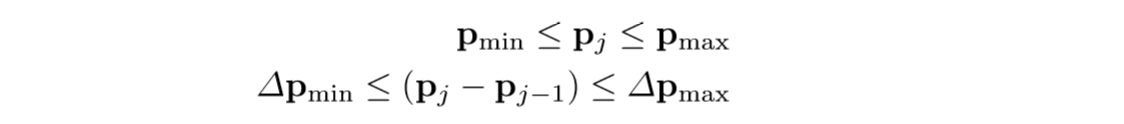


* ﬁx extra steps to initial pose:



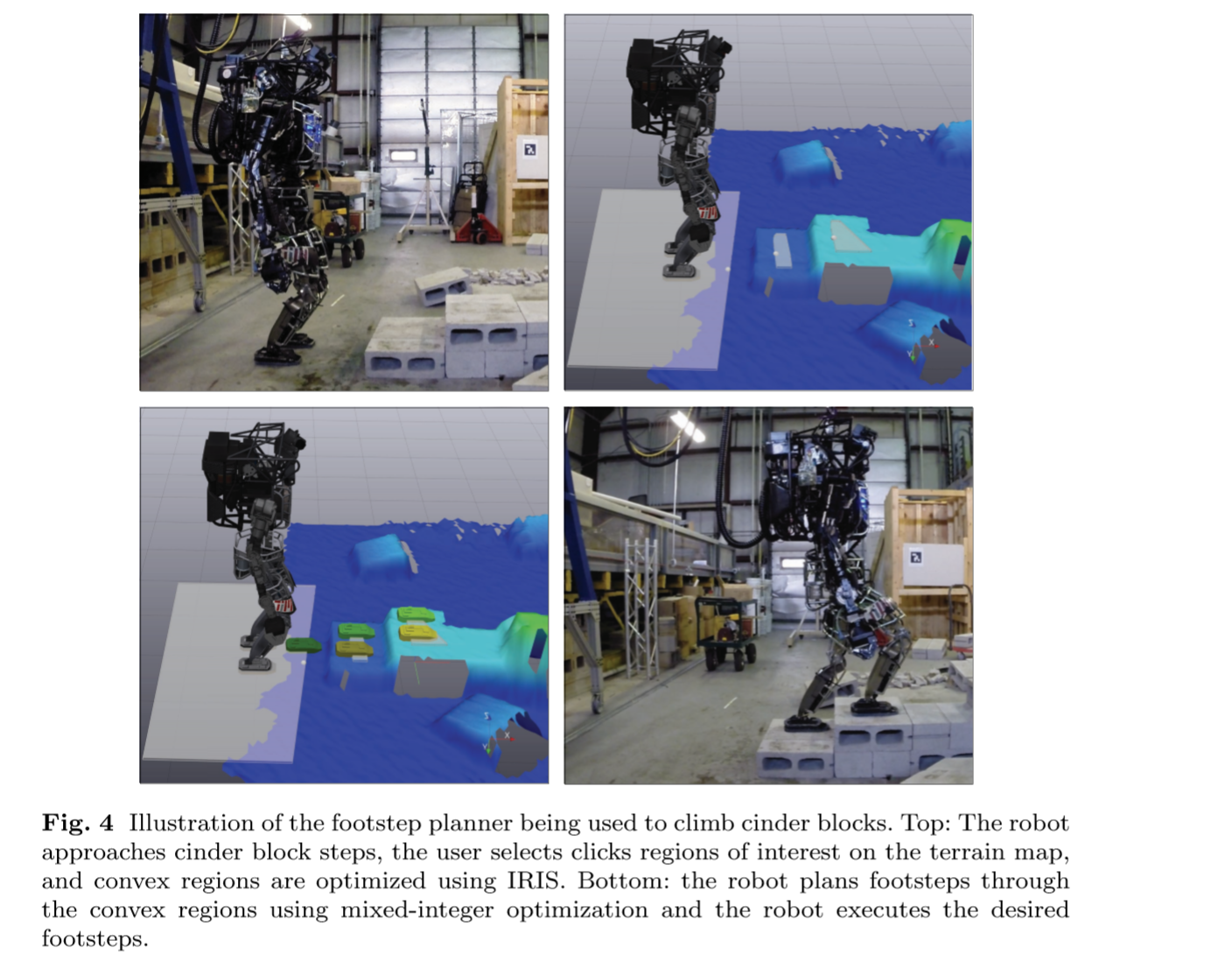
where p1 and p2 ﬁx to the initial poses of the robot’s feet;

* bounds on step positions and diﬀerences:

where, pmin, pmax, ∆pmin, ∆pmax ∈ R4 are bounds on the absolute footstep positions and their diﬀerences, respectively.

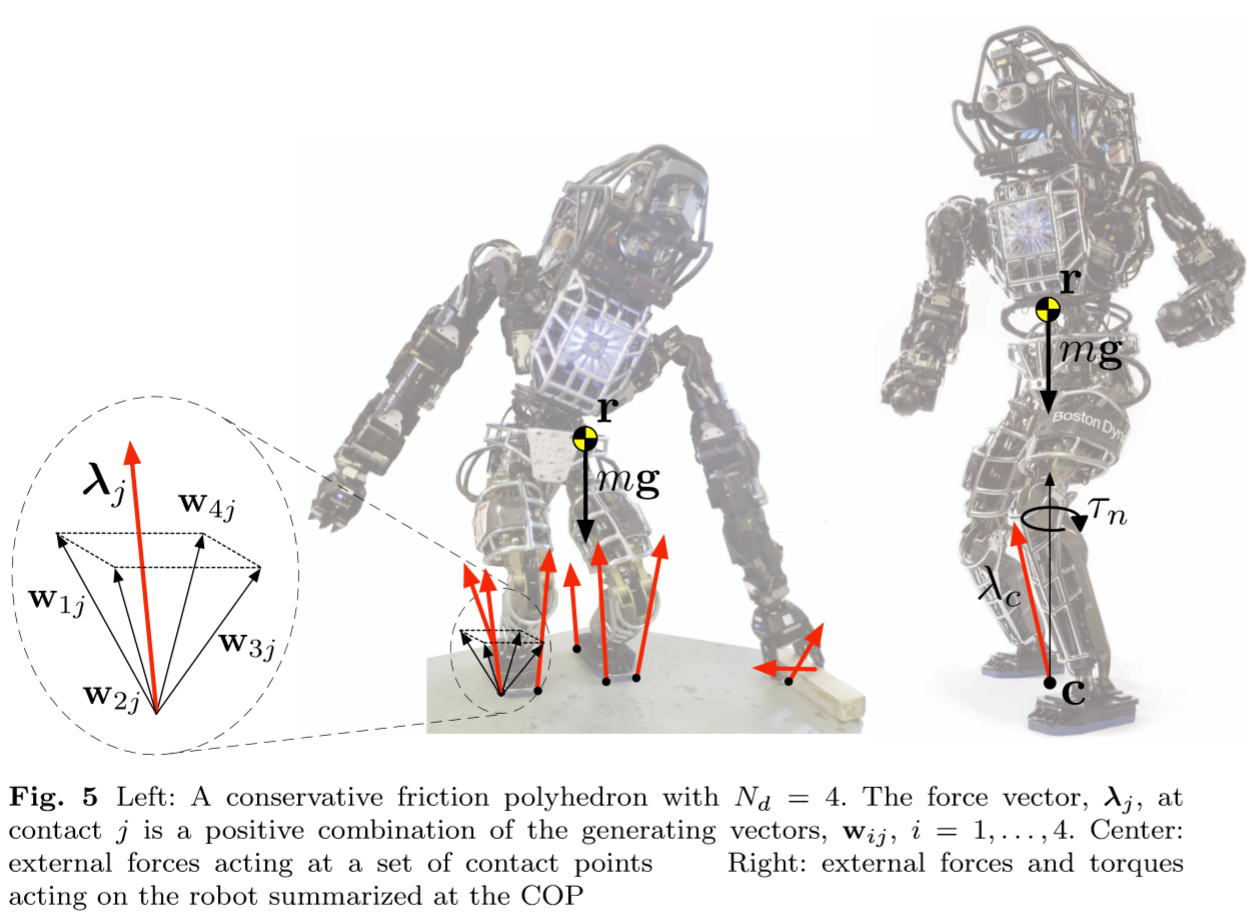
Despite the very large number of discrete decisions involved in solving the footstep planning problem to optimality, the mixed-integer convex formulation leads to extremely eﬃcient solutions in typical cases. For a footstep plan of N = 12 steps, in which each step must be assigned to one of R = 10 safe regions, L = 8 piecewise linearization of sin and cos, and 2 values of each ρj, there are, naively, 1012 × 812 × 212 ≈ 3 × 1026 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.

Given a desired footstep trajectory, a dynamic walking motion can be deﬁned using a piecewise polynomial center of pressure (COP) trajectory through the footsteps.



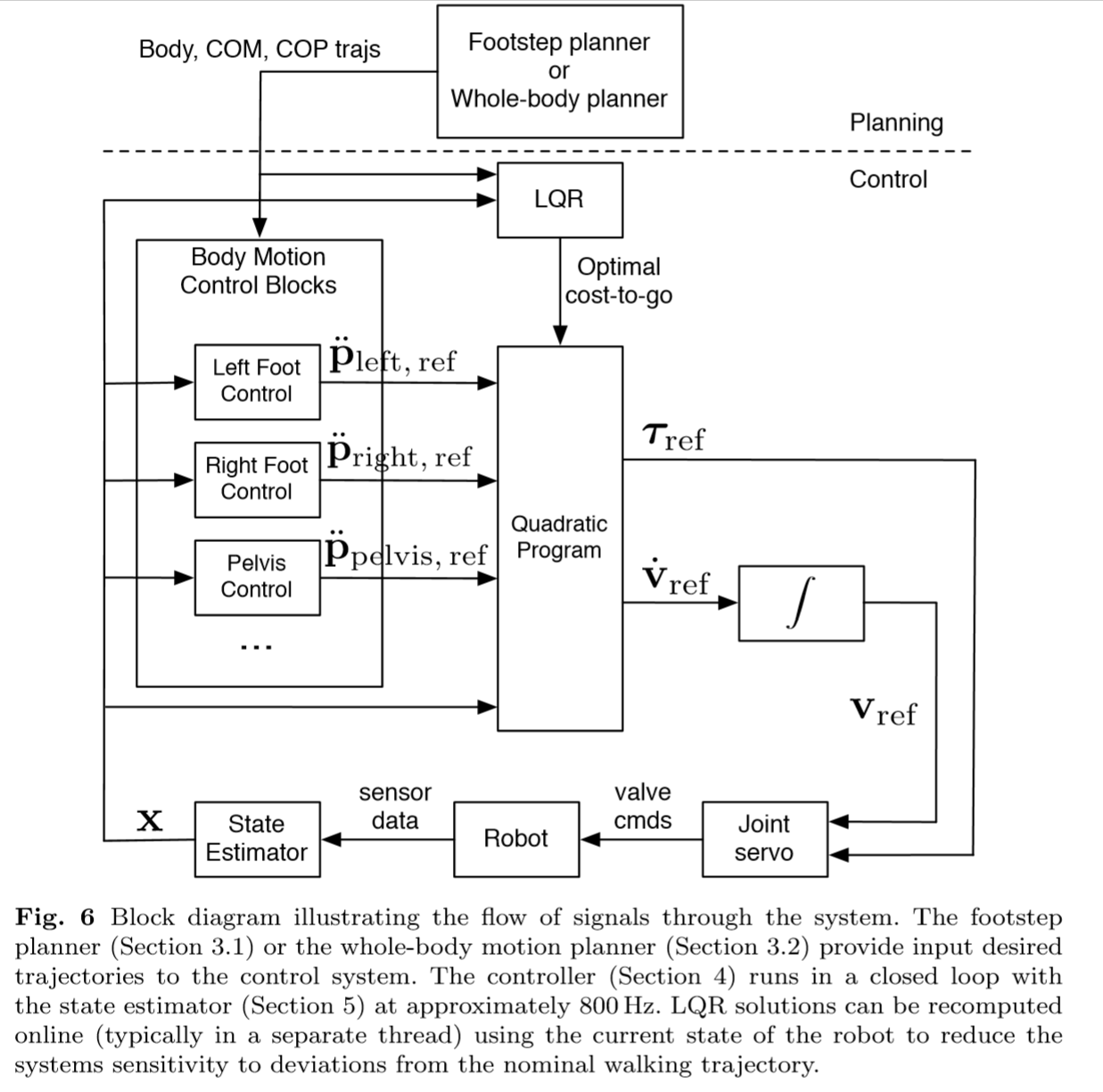
## Dynamic Motion Planning

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, 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, Boston Dynamics 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.



# 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.



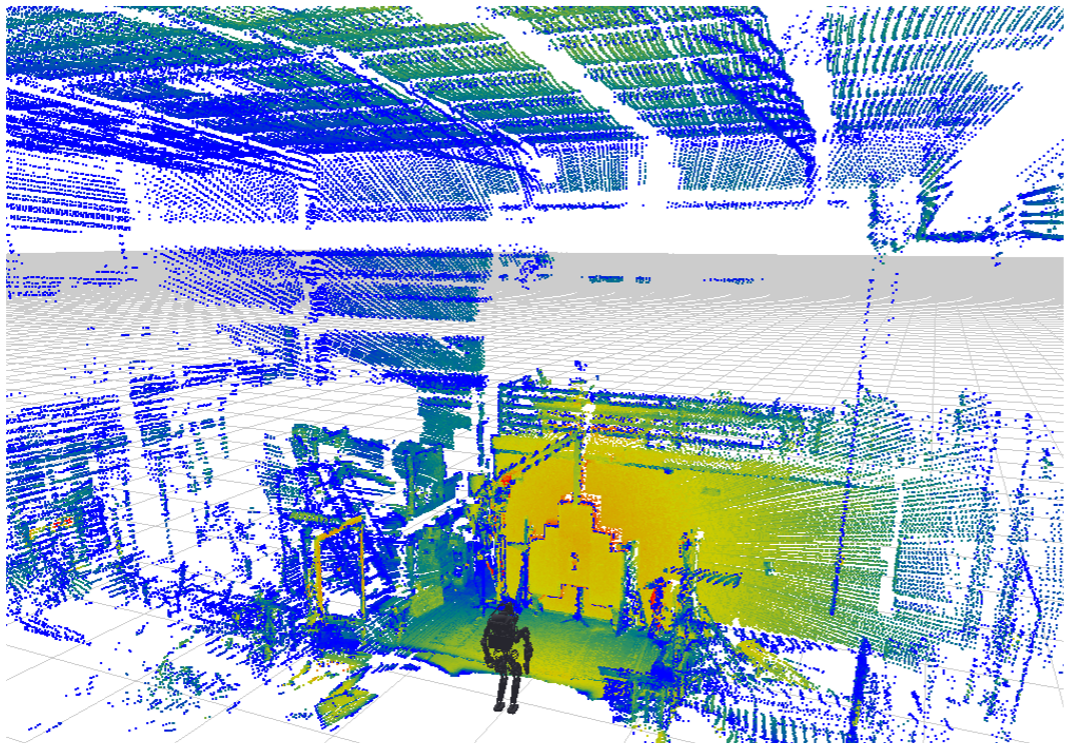
Requirements and approach

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. 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, a drift rate below 1cm per step is required. The proposed state estimator runs at 333Hz with a latency of 0.5msec 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.

# 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, nor can we aﬀord time to stop moving and perform static 3D registration, e.g., using an Iterative Closest Point algorithm. Instead we aim to incorporate information from each individual LIDAR scan into the state estimate using a Gaussian Particle Filter (GPF). In typical operation, the robot is ﬁrst commanded to stand still for about 30 seconds while it collects a full 3D point cloud of its environment (see Figure 7). This cloud is then converted into a probabilistic occupancy grid (OctoMap) against which eﬃcient localization comparisons later performed during locomotion. Actuated LIDAR with 30m range permits 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 the 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 1/40sec 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 2.5m at the highest spindle rotation speed. Accurate projection also requires precise calibration of the LIDAR sensor.



***Fig. 7*** The robot initially collects a static LIDAR point cloud of its environment, which is then converted into an occupancy map for subsequent localization.

# References

1. **(Full original text)** Scott Kuindersma · Robin Deits · Maurice Fallon · Andr´es Valenzuela · Hongkai Dai · Frank Permenter · Twan Koolen · Pat Marion · Russ Tedrake: Optimization-based Locomotion Planning, Estimation, and Control Design for the Atlas Humanoid Robot. Massachusetts Institute of Technology Cambridge, MA, USA; 2015
2. Boyd, S.P., Vandenberghe, L.: Convex optimization. Cambridge University Press, Cambridge, UK; New York (2004)
3. Fallon, M., Kuindersma, S., Karumanchi, S., Antone, M., Schneider, T., Dai, H., D’Arpino, C.P., Deits, R., DiCicco, M., Fourie, D., Koolen, T.T., Marion, P., Posa, 44 Scott Kuindersma et al. M., Valenzuela, A., Yu, K.T., Shah, J., Iagnemma, K., Tedrake, R., Teller, S.: An architecture for online aﬀordance-based perception and whole-body planning. J. of Field Robotics (2014)
4. Herdt, A., Diedam, H., Wieber, P.B., Dimitrov, D., Mombaur, K., Diehl, M.: Online walking motion generation with automatic foot step placement. Advanced Robotics 24(5-6), 719–737 (2010)
5. Xinjilefu, X., Feng, S., Huang, W., Atkeson, C.: Decoupled state estimation for humanoids using full-body dynamics. In: IEEE Intl. Conf. on Robotics and Automation (ICRA). Hong Kong, China (2014)
6. Bry, A., Bachrach, A., Roy, N.: State estimation for aggressive ﬂight in GPS-denied environments using onboard sensing. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1–8 (2012)
7. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte Carlo localization for mobile robots. In: IEEE Intl. Conf. on Robotics and Automation (ICRA) (1999)
8. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Machine Intell. 14(2), 239–256 (1992)
9. Deits, R., Tedrake, R.: Eﬃcient mixed-integer planning for UAVs in cluttered environments. In: IEEE Intl. Conf. on Robotics and Automation (ICRA) (2015)
10. Deits, R.L.H., Tedrake, R.: 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 (2014)