Motion planning for legged robots on varied terrain

Kris Hauser Timothy Bretl Jean-Claude Latombe
Kensuke Harada Brian Wilcox

July 31, 2007

Abstract

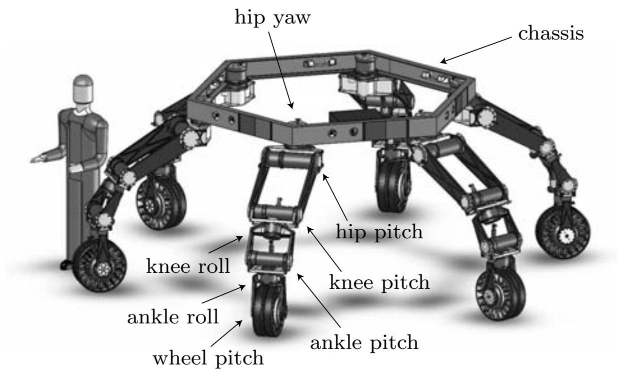
This paper studies the quasi-static motion of large legged robots that have many degrees of freedom. While gaited walking may suffice on easy ground, rough and steep terrain requires unique sequences of footsteps and postural adjustments specifically adapted to the terrain's local geometric and physical properties. This paper presents a planner to compute these motions. The planner combines graph searching techniques to generate a sequence of candidate footfalls with probabilistic sample-based planning to generate continuous motions that reach these footfalls. To improve motion quality, the probabilistic sample-based planner derives its sampling strategy from a small set of motion primitives that have been generated offline. The viability of this approach is demonstrated in simulation for the six-legged lunar vehicle ATHLETE and the humanoid HRP-2 on several example terrains, even one that requires both hand and foot contacts and another that requires rappelling.

1 Introduction

One of the main potential advantages of legged robots over other types of mobile robots (such as wheeled and track robots) is their mechanical ability to navigate on varied terrain. However, so far this ability has not been fully exploited. One reason is the lack of an adequate motion planner capable of computing unique sequences of footsteps and postural adjustments specifically adapted to the local geometric and physical properties of the terrain. In this paper we describe the design and implementation of a motion planner that enables legged robots with many degrees of freedom to navigate safely across varied terrain. Although most of this planner is general, our presentation focuses on two robots in particular: the six-legged lunar vehicle ATHLETE [70] and the humanoid HRP-2 [32].

1.1 ATHLETE and HRP-2

ATHLETE (shown in Fig. 1) is large and highly mobile. A half-scale Earth test model has diameter 2.75m and mass 850kg. It can roll at up to 10km/h on rotating wheels over flat smooth terrain and walk carefully on fixed wheels over irregular and steep terrain. With its



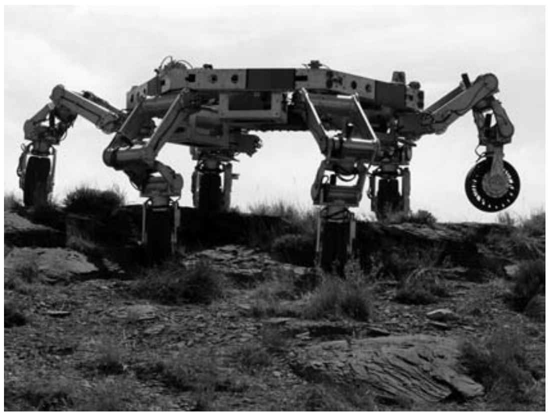


Figure 1: The ATHLETE lunar vehicle [70].

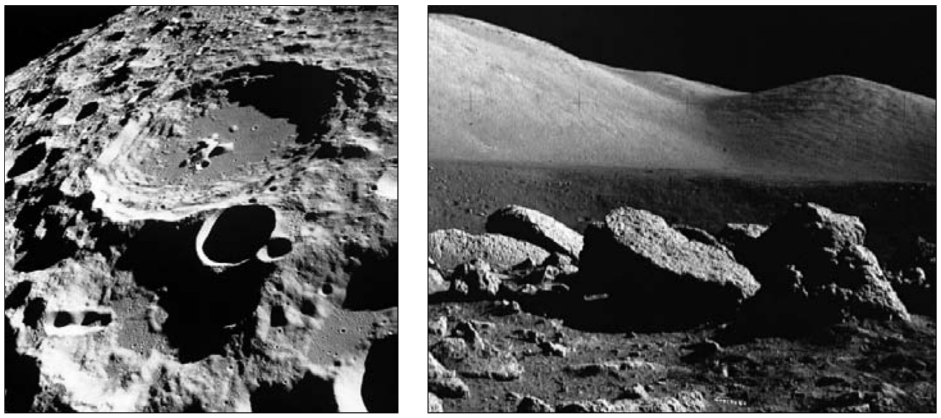


Figure 2: Pictures of lunar terrain from Apollo missions [27].

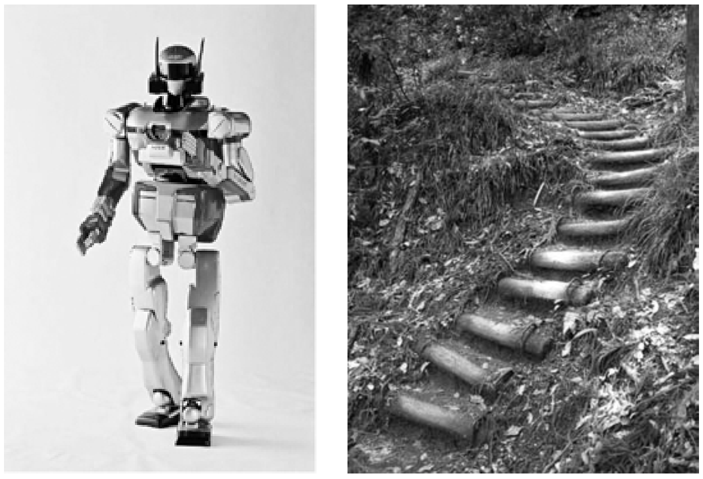


Figure 3: The humanoid HRP [32] and an example of varied terrain.

six articulated legs, ATHLETE is designed to scramble across terrain so rough that a fixed gait (for example, an alternating tripod gait) may prove insufficient. Such terrain is abundant on the Moon, most of which is rough, mountainous, and heavily cratered – particularly in the polar regions, a likely target for future surface operations. These craters can be of enormous size, filled with scattered rocks and boulders of a few centimeters to several meters in diameter (Fig. 2). Crater walls are sloped at angles of between 10-45°, and sometimes have sharp rims [27].

In comparison, HRP-2 (Fig. 3) is relatively light and compact, with height 1.5m and mass 50kg, and is capable of moving at up to 4km/h. Using a fixed gait, it can walk on flat surfaces, along narrow paths, and up stairs of constant height. It can even crawl through tunnels, hold onto handrails, and get up after falling down [25, 32]. But like ATHLETE, HRP-2 is potentially capable of scrambling across rougher terrain, such as irregular outdoor terrain or urban rubble, where a fixed gait may be insufficient.

Other robots and applications. A variety of previous work has also focused on enabling robots to traverse irregular terrain. Locomotion of humanoids across somewhat uneven terrain has been studied in [39, 75]. Other legged robots including quadrupeds [28], hexapods [65, 37], parallel walkers [74], and spherically symmetric robots [53], are capable of walking across rougher terrain. Wheeled robots with active or rocker-bogie suspension can also traverse rough terrain by changing wheel angles and center of mass position [17, 31, 42]. Careful descent is possible by rappelling as well, using either legs [4, 29, 69] or wheels [49]. The terrain we consider for ATHLETE and HRP-2 is more irregular and steep than in most previous applications, although not as steep as for free-climbing robots [9].

Careful walking also resembles dexterous manipulation. Both ATHLETE and HRP-2 grasp the terrain like a hand grasps an object, placing and removing footfalls rather than finger contacts. Legged robots have to remain in equilibrium as they move (only the object must remain in equilibrium during manipulation), and use fewer contact modes while walking (no sliding or rolling), but still face similar challenges [6, 52]. Manipulation planning, involving the rearrangement of many objects with a simple manipulator, is another related application. A manipulator takes a sequence of motions with and without a grasped object (different states of contact) just like a legged robot takes a sequence of steps [2]. In fact, for a legged robot to navigate among movable obstacles, it may be necessary to consider both walking and manipulation together [66].

1.2 Motion planning for legged robots

On rough terrain, the walking motion of legged robots like ATHLETE and HRP-2 is governed largely by two interdependent constraints: *contact* – i.e., keep feet, or fixed wheels, at a carefully chosen set of footfalls; and *equilibrium* – i.e., apply forces at these footfalls that exactly compensate for gravity without causing slip. The range of forces that may be applied at the footfalls without causing slip depends on their geometry (e.g., average slope) and their physical properties (e.g., coefficient of friction), both of which vary across the terrain. So every time the robot plans a step, it faces a dilemma: it can't know the constraints on its subsequent motion until it chooses a footfall, a choice that itself depends on the constraints.

To handle this dilemma in our planner, we make a key design choice similar to one introduced in [9] and [26] (Section 2): *to choose footfalls before computing motions.* We begin by identifying a number of potentially useful footfalls across the terrain. Each mapping of a robot's feet to a set of footfalls is a *stance*, associated with a (possibly empty) set of feasible configurations that satisfy all motion constraints (including contact and equilibrium). A robot can take a step from one stance to another if they differ by a single footfall and if they share some feasible configuration, which we call a *transition*. Our planner proceeds in two stages: first, we generate a candidate sequence of footfalls by finding transitions between stances; then, we expand this sequence into a feasible, continuous trajectory by finding paths between subsequent transitions. This two-stage planning approach is motivated by the fact that a legged robot's motion on irregular and steep terrain is most constrained just as it places a foot at or removes a foot from a footfall (more generally, when it makes a new contact or breaks one). At this instant, the robot must be able to reach the footfall (contact) but can not use it to avoid falling (equilibrium), since the applied force is zero when the contact is made or broken. So transitions are the "bottlenecks" of any motion: if we can find two subsequent transitions, it is likely we can find a path between them. This statement has been verified in our experiments.

Like the planners in [9] and [26], our planner combines graph searching to generate a sequence of candidate footfalls with a variant of the Probabilistic-Roadmap (PRM) approach (see Chap. 7 of [14]) to generate continuous motions that reach these footfalls. But, we add two key algorithmic tools in this framework (Section 3) to deal specifically with difficult computational issues raised by legged robots like ATHLETE and HRP-2. One is a method of sampling feasible configurations (both from scratch and by perturbation) and of connecting pairs of configurations with local paths. This method addresses the challenge that these robots have many degrees of freedom with terrain contacts that close many kinematic chains. While the closure constraint reduces the robot's feasible space at a given stance to a submanifold of the robot's configuration space, other constraints (such as equilibrium) restrict the feasible space further to a subset of small volume inside that manifold. Hence, sampling feasible configurations and connecting them with feasible paths is particularly difficult and potentially time consuming. The other tool is a powerful heuristic to generate footfalls and guide our search through the highly combinatorial collection of stances. This heuristic addresses the challenge that moving across varied, but not extreme, terrain requires footfalls to be properly selected, while the number of candidate stances is enormous.

Previous work. Motion planning for legged robots requires computing both a sequence of footfalls and continuous motions that reach these footfalls. Previous planners differ primarily in which part of the problem they consider first:

- *Motion before footfalls.* When it does not matter much where a robot contacts its environment, it makes sense to compute the robot's (or object's) overall motion first. For example, a manipulation planner might generate a trajectory for the grasped object ignoring manipulators, then compute manipulator trajectories that achieve necessary re-grasps [34]. Similarly, a humanoid planner might generate a 2-D collision-free path of a bounding cylinder, then follow this path with a fixed, pre-defined gait [38, 54]. A related strategy is to plan a path for the robot's center of mass, then to compute footfalls and limb motions that keep the center of mass stable along this path [16]. Some planners even avoid reasoning about footfalls entirely [44]. These techniques are fast, but do not extend well to irregular and steep terrain.

- *Footfalls before motion.* When the choice of contact location is critical, it makes sense to compute a sequence of footfalls first. This approach is related to the work on manipulation planning presented in [2], which expresses connectivity between different states of contact as a graph. For "spider-robots" with zero-mass legs walking on horizontal terrain, the exact structure of this graph can be computed quickly using analytical techniques [8]. For more general systems, the graph can sometimes be simplified by assuming partial gaits, for example restricting the order in which limbs are moved [62] or restricting footsteps to a discrete set [39]. But when motion is distinctly non-gaited (as in manipulation planning [51, 58], free-climbing [9], or for ATHLETE and HRP-2 on varied terrain), each step requires the exploration of a distinct configuration space. This fact motivates the two-stage search strategy we adopt in Section 2.

1.3 Improving motion quality

Without additional consideration for motion quality, the above approach often generates motion that looks unnatural and inefficient. The reason is that, while robots like ATHLETE and HRP-2 have many degrees of freedom (DOF), we do not know in advance which of these DOF are actually useful, nor how many contacts may be needed. In some cases, there might exist too many feasible motions. On easy terrain like flat ground or stairs of constant height, the motion of a legged robot is lightly constrained, so that most of its DOF are unnecessary, and only feet need contact the ground. For example, although crawling would also be feasible for HRP-2 on flat ground, we would rather see the humanoid walk upright. Alternatively, on hard terrain like steep rock or urban rubble, the robot's motion is highly constrained. In this case, most of its DOF are essential and additional contacts (hands, knees, shoulders) might be required for balance. On varied terrain between these two extremes, the number of relevant DOF and the types of required contacts may change from step to step. Since our basic planner always considers all DOF (in order to find a feasible motion whenever one exists) it may then generate needless motions of arms or other DOF that are not required for balance, or that may achieve balance in clearly sub-optimal ways. Eliminating such motions in post-processing turns out to be particularly hard.

To solve this problem, we provide our planner with a small set of high-quality *motion primitives*, similar to [39, 73]. These motion primitives might include a step on flat ground, a step up a staircase, or a step on sloped terrain with a hand contact on a rock for balance. Such primitives may be designed by hand, produced by off-line precomputation (for instance, using optimization techniques), or extracted from captured motions of humans or animals.

We record each motion primitive as a nominal path through the robot's configuration space. Then, instead of sampling across all of configuration space to find transitions between stances and paths between transitions, our planner samples a growing distribution of configurations around the nominal path associated with a chosen motion primitive. Our simulation results demonstrate not only a marked increase in motion quality[1] for legged robots walking on varied terrain, but also a reduction in planning time. In the absence of a relevant primitive, the planner falls back on its general sampling method.

Other ways to improve motion quality. The usual way to improve motion quality is to post-process feasible motions using methods like "short-cut" heuristics [33, 64] and gradient descent algorithms [67, 21]. We use similar methods in our planner, but for legged robots it is difficult to eliminate all needless motions in post-processing. For this reason, motion primitives and other types of maneuvers have been applied widely to robotics and digital animation for legged robots (and other vehicles with complex dynamics). Four general strategies have been used:

- *Record and playback.* This strategy restricts motion to a library of maneuvers. Natural-looking humanoid locomotion on mostly flat ground can be planned as a sequence of precomputed feasible steps [39]. Robust helicopter flight can be planned as a sequence of feedfoward control strategies (learned by observing skilled human operators) to move between trim states [20, 19, 50, 18]. Robotic juggling can be planned as a sequence of feedback control strategies [12]. The motion of peg-climbing robots can be planned as a sequence of actions like "grab the nearest peg" [5]. In these applications, a reasonably small library of maneuvers is sufficient to achieve most desired motions. For legged robots on varied terrain, such a library may grow to impractical size.

- *Warp, blend, or transform.* Widely used for digital animation, this strategy also restricts motion to a library of maneuvers, but allows these maneuvers to be superimposed or transformed to better fit the task at hand. For example, captured motions of human actors can be "warped" to allow characters to reach different footfalls [71] or "retargetted" to control characters of different morphologies [22]. Of course, for a digital character it is most important to look good while for a legged robot it is most important to satisfy hard motion constraints. So although some techniques have been proposed to transform maneuvers while maintaining physical constraints [56, 63], this strategy seems better suited for animation than robotics.

- *Model reduction.* This strategy plans overall motion first, following this motion with a concatenation of primitives. For example, another way to generate natural-looking humanoid locomotion on flat ground is to approximate the robot as a cylinder, plan a 2-D collision-free path of this cylinder, and follow this path with a fixed gait [38,

[1]Exactly how motion quality should be measured is an open question, beyond the scope of this paper. Here, we define quality as inversely proportional to a linear combination of path length and sum-squared distance from an upright posture.

35, 54, 36]. A similar method is used to plan the motion of nonholonomic wheeled vehicles [41, 40]. A related strategy plans the motion of key points on a robot or digital actor (such as the center of mass or related ground reference points [55]), tracking these points with an operational space controller [61]. These approaches work well when it does not matter much where a robot or digital actor contacts its environment. When the choice of contact location is critical, as is often the case for legged robots on varied terrain, it makes more sense to compute a sequence of footfalls first.

- *Bias inverse kinematic solutions.* Like model reduction, this strategy first plans the motion of key points on a robot or digital actor, such as the location of hands or feet. But instead of a fixed controller, a search algorithm is used to compute a pose of the robot or actor at each instant that tracks these points (an inverse kinematic solution). One approach is to choose an inverse kinematic solution according to a probability density function learned from high-quality example motions [73, 24, 46, 47]. The set of examples give the resulting pose a particular "style." In fact, we take a similar approach in this paper, planning steps for a legged robot by sampling waypoints in a growing distribution around high-quality nominal paths.

1.4 Organization of this paper

Say what this paper does *not* do. For example, we do not consider dynamics, closed-loop control, visual feedback, robustness to errors (and error recovery), etc.

2 Design of the motion planner

Our planner extends a similar one for humanoid robots [26], which was based on earlier work for a free-climbing robot [9]. Here, we summarize our basic approach.

2.1 Motion constraints

A *configuration* of either ATHLETE or HRP-2, denoted q, is a parameterization of the robot's placement in 3-D space. For ATHLETE, q consists of 6 parameters defining the position and orientation of the robot's hexagonal chassis and a list of 36 joint angles (each leg has six actuated, revolute joints). For HRP-2, q consists of 6 parameters defining the position and orientation of the torso and a list of 30 joint angles. The set of all such q is the *configuration space*, denoted $\mathcal{Q}$, of dimensionality 42 for ATHLETE and 36 for HRP-2.

We consider terrain that might include a mixture of flat, sloped, or rocky ground. We assume that this terrain and all robot links are perfectly rigid. We also assume that we are given in advance a set of links that are allowed to touch the terrain. For ATHLETE, this set includes only its six wheels, to which brakes are applied (to prevent rolling) while the robot is walking. For HRP-2, this set might include hands, feet, or knees. We call the placement of a link on the terrain a *contact*, and fix the position and orientation of the link while the contact is maintained. We call a set of simultaneous contacts a *stance*, denoted by σ. Consider a stance σ with $n \geq 1$ contacts. The *feasible space* $\mathcal{F}_\sigma$ is the set of all feasible configurations of the robot at σ. To be in $\mathcal{F}_\sigma$, a configuration q must satisfy several constraints:

- *Contact.* The n contacts form a linkage with multiple closed-loop chains. So, q must satisfy inverse kinematic equations. Let $\mathcal{Q}_\sigma \subset \mathcal{Q}$ be the set of all configurations q that satisfy these equations. This set $\mathcal{Q}_\sigma$ is a sub-manifold of $\mathcal{Q}$ of dimensionality $42 - 6n$ for ATHLETE and $36 - 6n$ for HRP-2, which we call the *stance manifold.* This manifold is empty if it is impossible for the robot to achieve the contacts specified by σ, for example if two contact points are farther apart than the maximum span of two legs.

- *Static equilibrium.* To remain balanced, both ATHLETE and HRP-2 must be able to apply forces at contacts in σ that compensate for gravity without slipping. For valid forces to exist, the robot's center of mass (CM) must lie above its *support polygon.* But on irregular and steep terrain, the support polygon does not always correspond to the base of the robot's feet. For example, both ATHLETE and HRP-2 will slip off a flat and featureless slope that is too steep, regardless of their CM position. To compute the support polygon, we model each contact as a frictional point. Let $r_1, \ldots, r_n \in \mathbb{R}^3$ be the position, $\nu_i \in \mathbb{R}^3$ be the normal vector, μ_i be the static coefficient of friction, and $f_i \in \mathbb{R}^3$ be the reaction force acting on the robot at each point. We decompose each force f_i into a component $\nu_i^T f_i \nu_i$ normal to the terrain surface (in the direction ν_i) and a component $(I - \nu_i \nu_i^T) f_i$ tangential to the surface. Let $c \in \mathbb{R}^3$ be the position of the robot's CM (which varies with its configuration). Assume the robot has mass m, and the acceleration due to gravity is $g \in \mathbb{R}^3$. All vectors are defined with respect to a global coordinate system with axes e_1, e_2, e_3, where $g = -\|g\| e_3$. Then the robot is in static equilibrium if

$$\sum_{i=1}^{n} f_i + mg = 0 \qquad \text{(force balance)} \qquad (1)$$

$$\sum_{i=1}^{n} r_i \times f_i + c \times mg = 0 \qquad \text{(torque balance)} \qquad (2)$$

$$\|(I - \nu_i \nu_i^T) f_i\|_2 \leq \mu_i \nu_i^T f_i \text{ for all } i = 1, \ldots, n. \qquad \text{(friction cones)} \qquad (3)$$

These constraints are jointly convex in $f_1, \ldots, f_n$ and c. In particular, (1)-(2) are linear and (3) is a second-order cone constraint. In practice we approximate (3) by a polyhedral cone, so the set of jointly feasible contact forces and CM positions is a high-dimensional polyhedron [11, 9, 10]. Finally, since

$$c \times mg = m\|g\| \begin{bmatrix} -c \cdot e_2 \\ c \cdot e_1 \\ 0 \end{bmatrix}$$

then (1)-(2) do not depend on $c \cdot e_3$ (the CM coordinate parallel to gravity), so the support polygon is the projection of this polyhedron onto the coordinates e_1, e_2. There are many ways to compute this projection and to test the membership of c. An approach that works well for our application is [10].

- *Joint torque limits.* The above equilibrium test assumes the robot is a rigid body, "frozen" at configuration q. In reality, to maintain q each joint must exert a torque, which in turn must not exceed a given bound. Let τ be the vector of all joint torques exerted by the robot. These torques must satisfy

$$\tau = G(q) - \sum_{i=1}^{n} J_i(q)^T f_i, \tag{4}$$

  where $G(q)$ is the generalized gravity vector and $J_i(q)$ is the Jacobian of the point on the robot touching r_i. Let $\|\cdot\|$ be a weighted L_∞ norm where $\|\tau\| < 1$ implies that each joint torque is within bounds. We check joint torque limits by computing contact forces that satisfy (1)-(4) with minimum $\|\tau\|$ (a linear program), and verify $\|\tau\| < 1$.

- *Collision.* In addition to satisfying joint angle limits, the robot must avoid collision with the environment (except at contact points) and with itself. We use techniques based on bounding volume hierarchies to perform collision checking, as in [23, 60].

2.2 Two-stage search

We assume both ATHLETE and HRP-2 move from one place to another by taking a sequence of *steps*. Each step is a continuous motion at a fixed initial stance that ends by making or breaking a single contact to reach a new stance. In particular, suppose the robot begins at a configuration $q \in \mathcal{F}_\sigma$ at an initial stance σ. Let σ' be a new stance that either adds one contact to σ or removes one contact from σ. Then a single step from σ to σ' is a feasible path in $\mathcal{F}_\sigma$ from the initial configuration q to some final configuration $q' \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$, which we call a *transition*, that is feasible at both σ and σ'.

We say that two stances are connected if the robot can take a step from one to the other. We encode necessary conditions for connectivity as a *stance graph*. Each node of this graph is a stance. Two nodes σ and σ' are connected by an edge if there is a transition between $\mathcal{F}_\sigma$ and $\mathcal{F}_{\sigma'}$. So the existence of an edge in the stance graph is a necessary condition for the robot to take a step from one stance to another. We encode both necessary and sufficient conditions for connectivity as a *transition graph*. Each node of this graph is a transition. Two nodes $q \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ and $q' \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma''}$ are connected by an edge if there is a continuous path between them in $\mathcal{F}_\sigma$. So the existence of an edge in the transition graph is a necessary and sufficient condition for the robot to take a step from one stance to another. The stance and transition graphs represent the connectivity of the robot's configuration space at coarse and fine resolutions, respectively.

Our planner interweaves exploration of the stance graph and the transition graph, based on the method of [9]. The algorithm EXPLORE-STANCEGRAPH searches the stance graph

```
EXPLORE-STANCEGRAPH($q_{\text{initial}}, \sigma_{\text{initial}}, \sigma_{\text{final}}$)
 1  $Q \leftarrow \{\sigma_{\text{initial}}\}$
 2  while $Q$ is nonempty do
 3      unstack a node $\sigma$ from $Q$
 4      if $\sigma = \sigma_{\text{final}}$ then
 5          construct a path $[\sigma_1, \ldots, \sigma_n]$ from $\sigma_{\text{initial}}$ to $\sigma_{\text{final}}$
 6          $i \leftarrow$ EXPLORE-TRANSITIONGRAPH($\sigma_1, \ldots, \sigma_n, q_{\text{initial}}$)
 7          if $i = n$ then
 8              return the multi-step motion
 9          else
10              delete the edge $(\sigma_i, \sigma_{i+1})$ from the stance graph
11      else
12          for each unexplored stance $\sigma'$ adjacent to $\sigma$ do
13              if FIND-TRANSITION($\sigma, \sigma'$) then
14                  add a node $\sigma'$ and an edge $(\sigma, \sigma')$
15                  stack $\sigma'$ in $Q$
16  return "failure"

EXPLORE-TRANSITIONGRAPH($\sigma_i, \ldots, \sigma_n, q$)
 1  $i_{\text{max}} \leftarrow i$
 2  for $q' \leftarrow$ FIND-TRANSITION($\sigma_i, \sigma_{i+1}$) in each component of $\mathcal{F}_{\sigma_i} \cap \mathcal{F}_{\sigma_{i+1}}$ do
 3      if FIND-PATH($\sigma_i, q, q'$) then
 4          $i_{\text{cur}} \leftarrow$ EXPLORE-TRANSITIONGRAPH($\sigma_{i+1}, \ldots, \sigma_n, q'$)
 5          if $i_{\text{cur}} = n$ then
 6              return $n$
 7          elseif $i_{\text{cur}} > i_{\text{max}}$ then
 8              $i_{\text{max}} = i_{\text{cur}}$
 9  return $i_{\text{max}}$
```

Figure 4: Algorithms to explore the stance graph and the transition graph.

(Fig. 4). It maintains a priority queue Q of nodes to explore. When it unstacks σ_{final}, it computes a candidate sequence of nodes and edges from σ_{initial}. The algorithm EXPLORE-TRANSITIONGRAPH verifies that this candidate sequence corresponds to a feasible motion by searching a subset of the transition graph (Fig. 4). It explores a transition $q \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ only if (σ, σ') is an edge along the candidate sequence, and a path between $q, q' \in \mathcal{F}_\sigma$ only if σ is a node along this sequence. We say that EXPLORE-TRANSITIONGRAPH has *reached* a stance σ_i if some transition $q \in \mathcal{F}_{\sigma_{i-1}} \cap \mathcal{F}_{\sigma_i}$ is connected to q_{initial} in the transition graph. The algorithm returns the index i of the farthest stance reached along the candidate sequence. If this is not σ_{final}, then the edge (σ_i, σ_{i+1}) is removed from the stance graph, and EXPLORE-STANCEGRAPH resumes exploration.

The effect of this two-stage search strategy is to postpone the generation of one-step paths (a costly computation) until after generating transitions. It works well because, as we mentioned in Section 1, both ATHLETE's and HRP-2's motion on irregular and steep terrain is most constrained just as either robot places or removes a foot. In our experiments we have observed that if we can find $q \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ and $q' \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma''}$, then a path between q and q' likely exists in $\mathcal{F}_\sigma$. For example, Section 3.4 will present experiments with ATHLETE on a variety of terrain. In these experiments, there was a 60%-75% chance of finding a feasible path between randomly sampled $q \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ and $q' \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma''}$. Moreover, even if we could find no feasible path from q to q', nearly 100% of the time we could find a feasible path from q to a different configuration in $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma''}$. So after sampling even a small number of transitions, we can be reasonably sure to also find a one-step path.

Two key tools are embedded in this framework (the subroutines FIND-TRANSITION and FIND-PATH, and a heuristic for ordering Q) that we discuss in the following section.

3 Tools to support the motion planner

3.1 Generating transitions

Both EXPLORE-STANCEGRAPH and EXPLORE-TRANSITIONGRAPH require the subroutine FIND-TRANSITION to generate transitions $q \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ between pairs of stances σ and σ'. To implement FIND-TRANSITION, we use a sample-based approach. The basic idea is to sample configurations randomly in $q \in \mathcal{Q}$ and reject them if they are not in $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$. But since $\mathcal{Q}_\sigma$ has zero measure in $\mathcal{Q}$, this approach will never generate a feasible transition. So like [68, 72, 15], we spend more time trying to generate configurations that satisfy the contact constraint at σ (hence, at σ' if $\sigma' \subset \sigma$) before rejecting those that do not satisfy other constraints. Like [26], we do this in two steps:

- *Create a candidate configuration that is close to $\mathcal{Q}_\sigma$.*

  This step is tailored to the particular legged robot:

  - ATHLETE: Each contact in the stance corresponds to the placement of a foot at a footfall in the terrain. First, we create a nominal position and orientation of the chassis: (1) given a stance σ, we fit a plane to the footfalls in a least-squares sense; (2) we place the chassis in this plane, minimizing the distance from each hip to its corresponding footfall; (3) we move the chassis a nominal distance parallel to the plane-fit and away from the terrain. Then, we sample a position and orientation of the chassis in a Gaussian distribution about this nominal placement. Finally, we compute the set of joint angles that cause each foot to either reach or come closest to reaching its corresponding footfall. Note that a footfall fixes the intersection of the ankle pitch and ankle roll joints relative to the chassis (Fig. 1). The hip yaw, hip pitch, and knee pitch joints determine this position. There are up to four inverse kinematic solutions for these joints – or, if no solutions exist, there are two configurations that are closest (straight-knee and completely bent-knee). The knee roll, ankle roll, and ankle pitch determine the orientation of the foot, for which there are two inverse kinematic solutions. We select a configuration that satisfies joint-limit constraints; if none exists, we reject the sample and repeat.

  - Hrp-2: Each contact in the stance corresponds to the placement of a robot link on the terrain. We select one of these links as a root and fix its location. Then, starting from the root link, we incrementally sample joint angles (satisfying joint angle limits) along each closed-loop kinematic chain using a bounding-bolume technique similar to [15]. Finally, we use cyclic coordinate descent as in [68] to adjust these joint angles so that every contact in the stance is approximately achieved.

- *Repair the candidate configuration using numerical inverse kinematics.* We move the candidate configuration to a point in $\mathcal{Q}_\sigma$ using an iterative Newton-Raphson method. For each contact, we can represent the error in position and orientation of the corresponding robot link i as a differentiable function $f_i(q)$ of the configuration q. Let

$$g(q) = \begin{bmatrix} f_1(q) \\ \vdots \\ f_n(q) \end{bmatrix}$$

so we can write the contact constraint as the equality $g(q) = 0$. Assume we are given a candidate configuration q_1. Then at each iteration k, we transform this configuration by taking the step

$$q_{k+1} = q_k - \alpha_k \nabla g(q_k)^{-\dagger} g(q_k),$$

where $\nabla g(q_k)^{-\dagger}$ is the pseudo-inverse of the gradient of the error function, and α_k is the step size (computed using backtracking line search). The algorithm terminates with success if at some iteration $\|g(q_k)\| < \varepsilon$ for some tolerance ε, or with failure if a maximum number of iterations is exceeded.

The first step rarely generates configurations in $\mathcal{Q}_\sigma$ but quickly generates configurations that are close to $\mathcal{Q}_\sigma$. Conversely, the primary cost of the second step is in computing $\nabla g(q_k)^{-\dagger}$ at every iteration, but if candidate configurations are sufficiently close to $\mathcal{Q}_\sigma$ then few iterations are necessary. So, it is the combination of these two methods that makes our sampler fast. For athlete, the experiments corresponding to Fig. 8 show that the repair step increases the fraction of feasible configurations from 1.9% to 18.4% and reduces the average time it takes to generate each feasible sample from 0.64s to 0.24s. For hrp-2, the experiments corresponding to Fig. 12 show that the repair step increases the fraction of feasible configurations from 0.4% to 31.9% and reduces the sampling time from 0.74s to 0.06s.

Our approach quickly samples configurations that satisfy the contact constraint. One problem with this approach is that as other constraints (such as equilibrium and joint limits) become more restrictive, it becomes less likely that a configuration in $\mathcal{Q}_\sigma$ will also be in $\mathcal{F}_\sigma$. This problem arises in particular for hrp-2, which in general has a smaller support polygon and tighter joint limits than athlete. To reduce the rejection rate, we write the equilibrium and joint-limit constraints as differentiable inequalities, and enforce them together with the

```
FREE-PATH($q, q'$)
1  if the distance from $q$ to $q'$ is less than $\varepsilon$ then
2      return TRUE
3  $q_{\text{mid}} \leftarrow (q + q')/2$
4  if Newton-Raphson from $q_{\text{mid}}$ results in $q_{\text{mid}} \in \mathcal{Q}_\sigma$ then
5      if $q_{\text{mid}} \in \mathcal{F}_\sigma$ then
6          return (FREE-PATH($q, q_{\text{mid}}$) & FREE-PATH($q_{\text{mid}}, q'$))
7      else
8          return FALSE
9  else
10     return FALSE
```

Figure 5: Algorithm to connect close configurations with a local path.

contact equality when we repair a candidate configuration. Including these inequalities requires a slight modification of the Newton-Raphson procedure (specifically, an active-set method as in [13]).

Note that EXPLORE-TRANSITIONGRAPH additionally requires that we sample a single transition in each connected component of $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$. Our approach is not guaranteed to do this, but the probability that it samples at least one in each component increases with the number of samples.

3.2 Generating paths between transitions

EXPLORE-TRANSITIONGRAPH requires the subroutine FIND-PATH to generate paths in $\mathcal{F}_\sigma$ between pairs of transitions $q \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ and $q' \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma''}$. We use a variant of the probabilistic roadmap approach called SBL that is bi-directional (growing trees from both q and q') and lazy (delaying the creation of local paths until a candidate sequence of milestones is found) [59].

To sample configurations in $\mathcal{F}_\sigma$, we face the same challenge discussed in the previous section (that a random configuration has zero probability of being in $\mathcal{Q}_\sigma$), and so we use a similar approach. However, in this case we can focus our search on a small part of feasible space, near existing milestones in each tree of the roadmap. Rather than sample a candidate configuration $q \in \mathcal{Q}$ at random, we sample it in a neighborhood of an existing configuration q_0. Close to q_0, the shape of $\mathcal{Q}_\sigma$ is approximated well by the hyperplane

$$\{\, p \in \mathcal{Q} \mid \nabla g(q_0)^T p = \nabla g(q_0)^T q_0 \,\}.$$

So before applying the iterative method to repair the sampled configuration, we first project it onto this hyperplane (as in [72]).

To connect milestones with local paths, we face a similar challenge, since the straight-line path between any two configurations q and q' will not (in general) lie in $\mathcal{Q}_\sigma$. So, we deform this straight-line path into $\mathcal{Q}_\sigma$ using the bisection method FREE-PATH (Fig. 5). At each iteration, FREE-PATH first applies Newton-Raphson (see Section 3.1) to the midpoint of q and q' to generate $q_{\text{mid}} \in \mathcal{Q}_\sigma$, then it checks that $q_{\text{mid}} \in \mathcal{F}_\sigma$. If both steps succeed, the algorithm continues to recurse until a desired resolution has been reached; otherwise, the algorithm returns failure. The advantage of this approach is that it does not require a direct local parameterization of $\mathcal{Q}_\sigma$, as it may be difficult to compute such a parameterization that covers both q and q'.

3.3 Ordering the graph search

Our two-stage search strategy can be improved by ordering the stances in Q according to a heuristic cost function $g(\sigma) + h(\sigma)$ in EXPLORE-STANCEGRAPH, where stances with lower cost are given higher priority. We define $g(\sigma)$ as the minimum number of steps required to reach σ from σ_{initial} in the stance graph. We define $h(\sigma)$ as a weighted sum of several criteria:

- *Planning time.* We increase the cost of σ proportional to the amount of time spent trying to sample a transition $q \in \mathcal{F}_{\sigma'} \cap \mathcal{F}_\sigma$ to reach it [51].

- *Distance to goal.* We increase the cost of σ proportional to the distance between the centroid of its contacts and those of the goal stance σ_{final}.

- *Footfall distribution.* We increase the cost of σ proportional to the difference (in a least-squares sense) between its contacts and those of a nominal stance on flat ground (for example, with feet directly under each hip).

- *Equilibrium criteria.* We increase the cost of σ inversely proportional to the area of its support polygon.

This heuristic reduces planning time and improves the resulting motion. It also allows us to relax an implicit assumption – that FIND-TRANSITION and FIND-PATH always return "failure" correctly. Because we implement these subroutines using a probabilistic, sample-based approach, we are unable to distinguish between impossible and difficult queries. So on failure of FIND-TRANSITION in EXPLORE-STANCEGRAPH, we still add σ to the stance graph but give σ a high cost. Likewise, rather than delete (σ, σ') on failure of FIND-PATH, we increase the cost of σ and σ'.

3.4 Implementation and Results

We tested the planner in simulation on several example terrains. It enables ATHLETE to traverse terrain that are otherwise inaccessible by gaits. Table 1 shows tests on a series of stair steps. The stairs range from 0.2 to 0.5 times the diameter of ATHLETE's chassis, and require moving about 2 body lengths. Alternating tripod, four-legged, and six-legged gaits were able to traverse the lowest stair (after some recoverable slippage), but failed on all others. The planner, however, was able to reliably plan all stairs. We randomly sample 200 footfalls in each terrain to use in the planner, relying on our graph search heuristic (Section 3.3) to

	Gait			Planner	Manual
Height	Tripod	Four	Six		
0.2	✓	✓	✓	8m	5m40s
0.3	—	—	—	8m30s	14m
0.4	—	—	—	16m15s	—
0.5	—	—	—	15m15s	—

Table 1: Stair steps planned with various methods. Dashes indicates failure. Planner times are averaged over four runs.

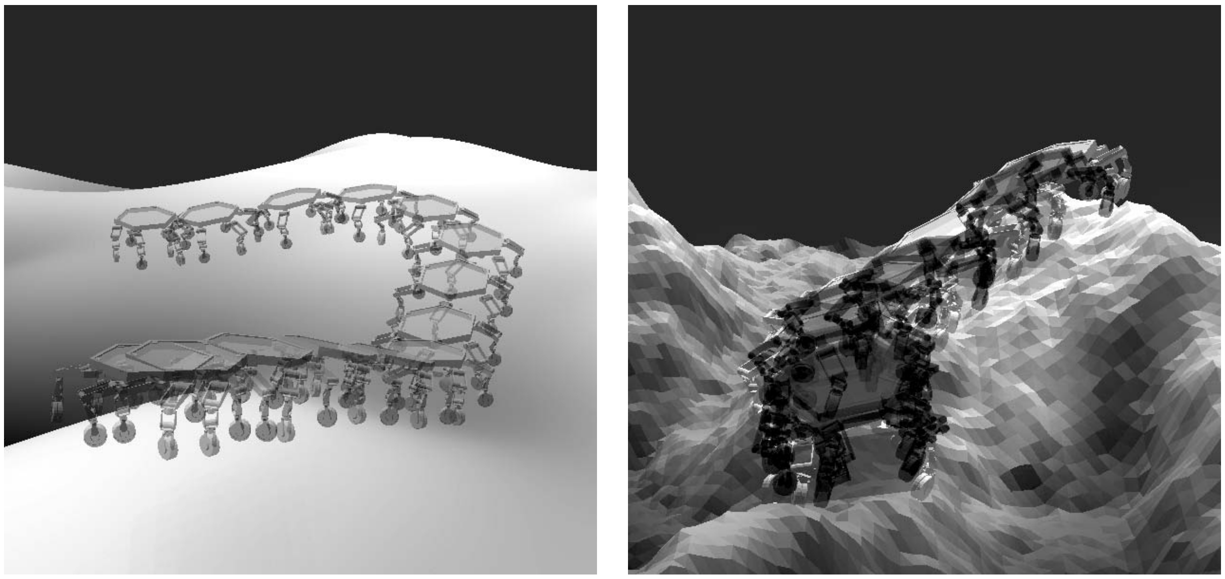


Figure 6: Walking with an alternating tripod gait is (a) feasible on smooth terrain but (b) infeasible on uneven terrain. Infeasible configurations are highlighted red.

identify which of these footfalls are useful. We are working on ways to better refine our selection of footfalls (for example, during incremental sensing), but right now the benefit is marginal. We compare the planner with footsteps chosen manually. A human operator used a point-and-click interface place and break contacts. Motions to achieve the commanded contact changes were planned automatically with the one-step planner. Manual operation was straightforward for the 0.2-unit stair, but the 0.3-unit stair required a large amount of trial-and-error and backtracking. An attempt to plan the 0.4-unit stair was stopped in frustration after about 30 minutes.

Next, we test the planner on terrains generated to simulate a range of lunar surfaces. Using a fractal generation method, we create height-maps of the form $z = f(x, y)$ as triangle mesh consisting of 32768 triangles, each about half the size of one of ATHLETE's wheels. All contacts are modeled with the same coefficient of friction. Fig. 6(a) shows an alternating-tripod gait applied to smooth, undulating terrain. The gait can traverse the terrain freely.

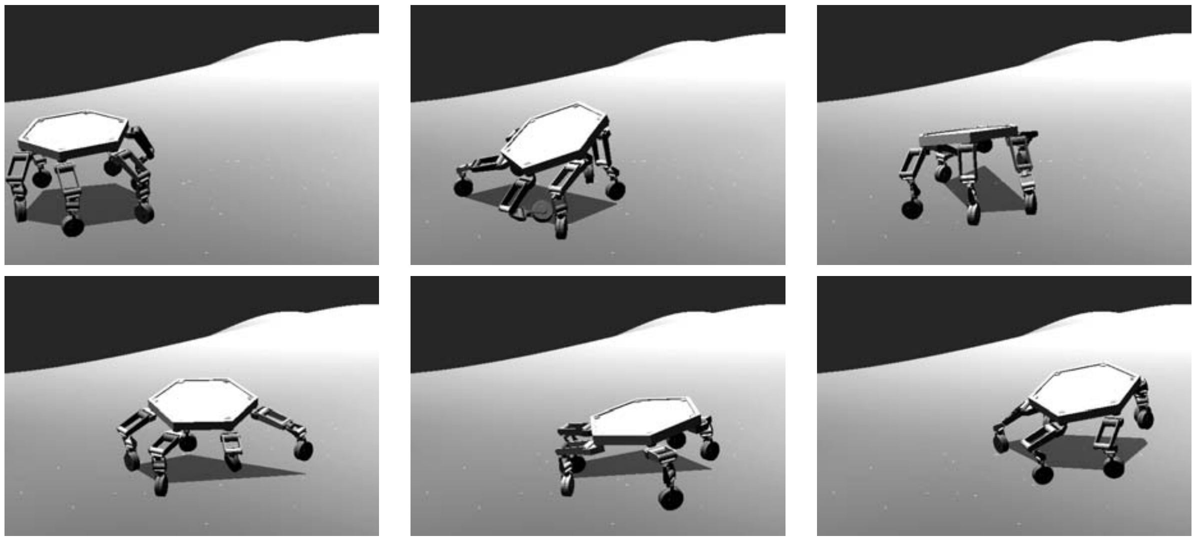


Figure 7: Walking on smooth, undulating terrain with no fixed gait.

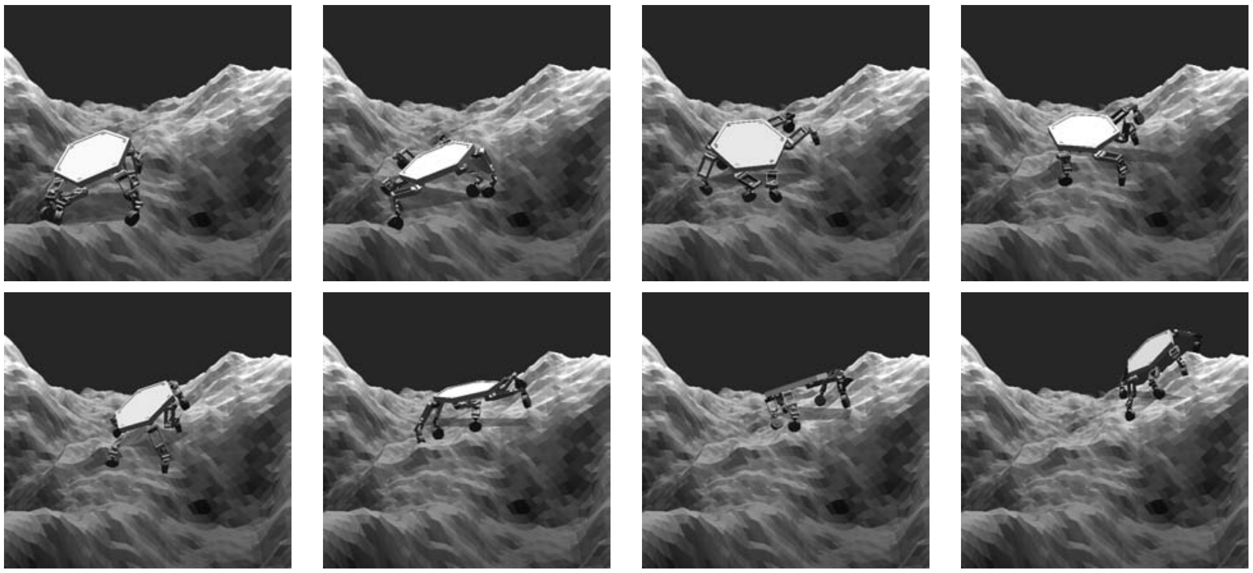


Figure 8: Walking on steep, uneven terrain with no fixed gait.

However, on irregular and steep ground, the gait does not work at all – it causes ATHLETE to lose balance or exceed torque limits at several locations (Fig. 6(b)). We apply our planner to the same terrains, setting the initial and final stances at a distance of about twice the diameter of ATHLETE's chassis, and sampling 200 contacts to use in the planner. Fig. 7 shows motion on smooth ground, computed in 14 minutes and consisting of 66 steps. Fig. 8 shows a feasible motion on irregular and steep ground, computed in 26 minutes and consisting of 84 steps.

The planner is also flexible enough to handle different robot morphologies. Fig. 9 shows

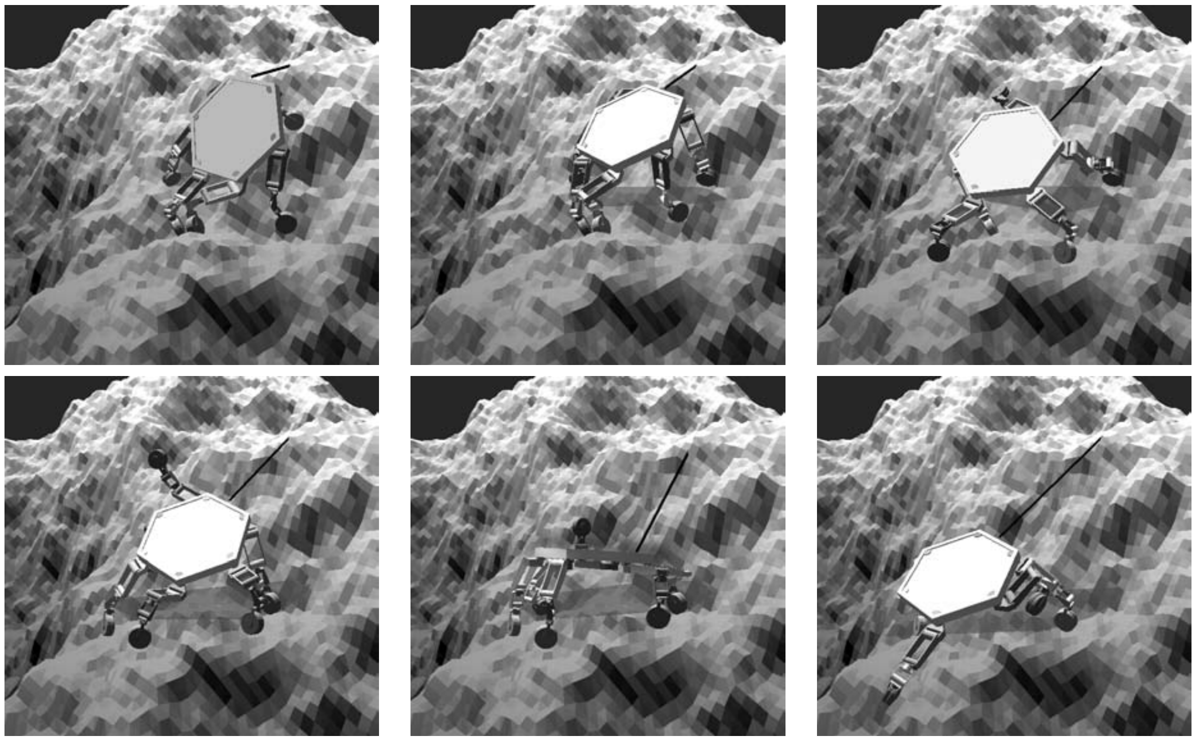


Figure 9: Rappelling down an irregular 60° slope with no fixed gait.

motion to descend irregular and steep terrain at an average angle of about 60°. In this example, ATHLETE is rappelling, using a tether (anchored at the top of the cliff) to help maintain equilibrium. We included the tether with no modification to our planner, treating it as an additional leg with a different kinematic structure. The resulting motion consisted of 32 steps. Total computation time was 16 minutes.

4 Improving motion quality

Because we use probabilistic sample-based methods to sample transitions and plan paths between them, the motions we generate are feasible (given an accurate terrain model) but not necessarily high-quality. In particular, when ATHLETE and HRP-2 walk on terrain that is not irregular and steep, their motion is lightly constrained. Each step we generate might contain strange or erratic motions of the arms and legs. To improve the result, we apply a method of smoothing similar to [21, 67], which uses gradient descent to achieve criteria like minimum path length and maximum clearance (or safety margin). However, this type of post-processing does not eliminate all needless motion.

Moreover, because we sample each contact, we might end up trying difficult steps when simpler ones would have led to the goal as well. For example, the robot might reach a stance σ associated with a feasible space $\mathcal{F}_\sigma$ containing a narrow passage. With only a small perturbation of the contacts at σ, this narrow passage is likely to disappear [30]. So although

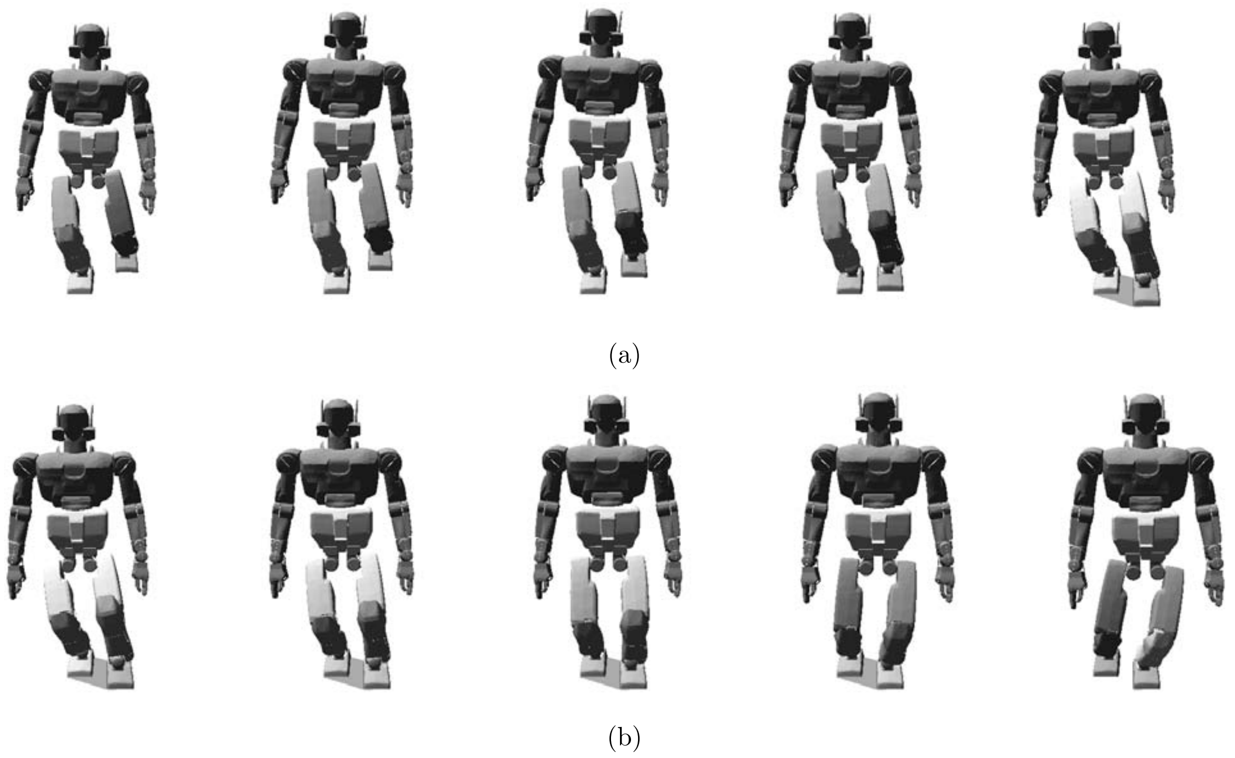


Figure 10: Two primitives on flat ground, to (a) place a foot and (b) remove a foot. The support polygon – here, just the convex hull of supporting feet – is shaded blue.

additional steps might still be possible, they would be easier to compute if we had made a better choice of contacts at σ.

4.1 Generating motion primitives

We address the limitations of our planner by using a library of *motion primitives*. Each primitive is a single step of very high quality. In this section, we describe how we generate primitives. In the following section, we will describe how they guide our selection of paths, transitions, and contacts.

Currently, it is the responsibility of the user to decide which primitives to include in the library. First, we need to identify a small but representative set of steps to be learned and to specify start and goal stances (differing by a single contact) for each one. These steps should be both important (often repeated) and broadly applicable (similar to a wide variety of other steps). For example, we might choose to include several consecutive steps on flat ground, each placing or removing a foot (Fig. 10). Next, we need to define a weighted set of criteria to judge the quality of each step. For example, we might choose to minimize path length, torque, energy, or the amount of deviation from an upright posture. Finally, we need to decide whether to accept or reject a candidate primitive, because we are not guaranteed that our optimization criteria correspond to our aesthetic notion of what is “natural.”

It is the responsibility of the planner to actually compute each primitive. First, we generate an initial trajectory between the given start and goal stances by randomly sampling a feasible transition and creating a path to reach it using PRM, as in [26, 9]. Then, we optimize this trajectory with respect to the given objective function using a standard nonlinear optimization package [43]. This entire process is an off-line precomputation; several hours were required to generate the two example primitives in Fig. 10.

The generation of motion primitives has not been the main focus of our work (here we are interested in their application), so many improvements may be possible. For example, we expect better results to be obtained by using the method of optimization proposed by [7]. Likewise, we might use a learned classifier to decide (without supervision) whether candidate primitives look natural, as in [57]. Finally, we might automate the selection of primitives to include in our library by learning a statistical model of importance (similar to location-based activity recognition [45]) or applicability after perturbation (similar to PRM planning with model uncertainty [48]).

We record each primitive in our library as a nominal path

$$u\colon t \in [0,1] \to u(t) \in \mathcal{Q}$$

in configuration space that does one of two things:

- *Adds a contact.* For some σ and σ' such that $\sigma \subset \sigma'$, u is a feasible path in $\mathcal{F}_\sigma$ from $u(0) \in \mathcal{F}_\sigma$ to $u(1) \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$.
- *Removes a contact.* For some σ and σ' such that $\sigma \supset \sigma'$, u is a feasible path in $\mathcal{F}_\sigma$ from $u(0) \in \mathcal{F}_\sigma$ to $u(1) \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$.

We will denote the start and goal stances for each primitive u by σ_u and σ'_u, respectively. In general, u will only define a feasible step between σ_u and σ'_u, but we will see in the next section that it can still be used to help guide our choice of path, transition, and contact to reach other stances.

4.2 Using primitives for planning

We use motion primitives to help our planner generate each step. We do this at three levels: finding a path (given a transition and a final stance), finding a transition (given only the final stance), and finding a contact (in order to define the final stance). In each case, first we transform the primitive to better match the step we are trying to plan, then we apply the transformed primitive to bias the sampling strategy used by our planner.

4.2.1 Finding paths

Consider the robot at an initial configuration $q_{\text{initial}} \in \mathcal{F}_\sigma$ at an initial stance σ. Assume that we are given a final stance σ' and a transition $q_{\text{final}} \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ (recall q_{final} is a configuration feasible at both σ and σ'). Also assume that we are given an appropriate primitive $u \subset \mathcal{Q}$ (as described in Section 4.1). We want to use u to guide our search for a path from q_{initial}

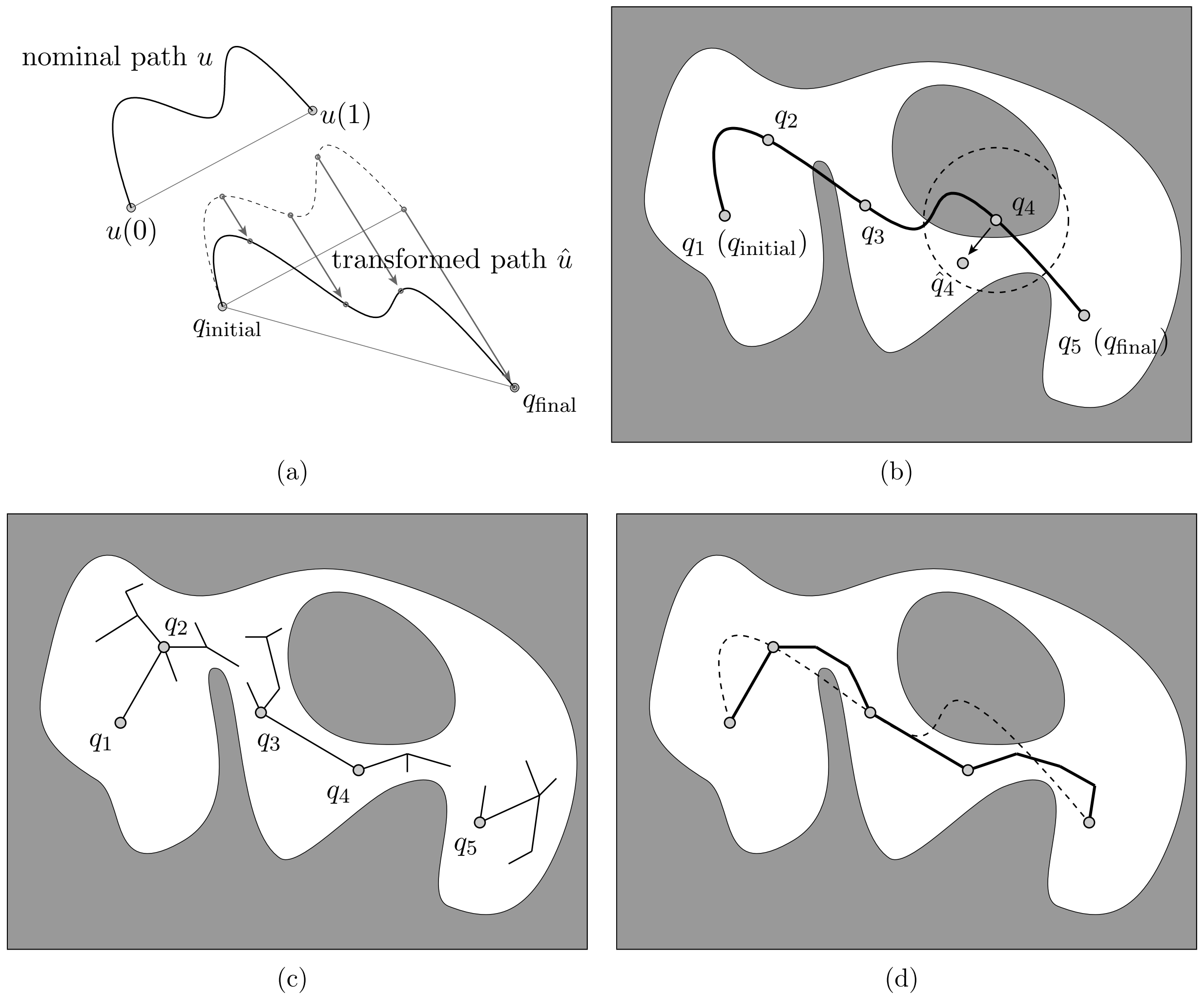


Figure 11: Using a primitive to guide path planning. (a) Transforming a motion primitive to start at q_{initial} and end at q_{final}. (b) Sampling root milestones in $\mathcal{F}_\sigma$ near equally spaced waypoints along $\hat{u}$. (c) Growing trees to connect neighboring roots. (d) The resulting path, which if possible is close to $\hat{u}$ (dotted).

to q_{final} in $\mathcal{F}_\sigma$. As before, we use SBL (a variant of PRM) to grow trees from root configurations [59]. But rather than root these trees only at q_{initial} and q_{final}, we root them at additional configurations (similar to [1]) sampled according to the primitive u.

Transforming the primitive to match q_{initial} and q_{final}. Although we assume u is similar to the step we are trying to plan, it will not be identical. So first, we transform u so that it starts at q_{initial} and ends at q_{final}. We have chosen to use an affine transformation of the form

$$\hat{u}(t) = A\left(u(t) - u(0)\right) + q_{\text{initial}} \tag{5}$$

that maps the straight-line segment between $u(0)$ and $u(1)$ to the segment between q_{initial} and q_{final}. In other words,

$$\begin{aligned} \hat{u}(0) &= A\left(u(0)-u(0)\right)+q_{\text{initial}} & \qquad \hat{u}(1) &= A\left(u(1)-u(0)\right)+q_{\text{initial}} \\ &= 0+q_{\text{initial}} & &= \left(q_{\text{final}}-q_{\text{initial}}\right)+q_{\text{initial}} \\ &= q_{\text{initial}} & &= q_{\text{final}} \end{aligned}$$

In particular, we select A closest to the identity matrix, minimizing

$$\min_{A} \sum_{i,j} (A_{ij}-\delta_{i,j})^2 \text{ such that } A\left(u(1)-u(0)\right) = q_{\text{final}}-q_{\text{initial}}$$

where $\delta_{ij}=1$ if $i=j$ and 0 otherwise. We compute A in closed form as

$$A = I + \frac{\left(\left(q_{\text{final}}-q_{\text{initial}}\right)-\left(u(1)-u(0)\right)\right)\left(u(1)-u(0)\right)^T}{\|u(1)-u(0)\|_2^2}.$$

We can visualize this transformation as in Fig. 11(a). First, u is translated to start at q_{initial}. Then, the farther we move along u (the more we increase t), the closer $\hat{u}$ is pushed toward the segment from q_{initial} to q_{final}.

Sampling root milestones. Let $q_1, \ldots, q_n$ be configurations evenly distributed along $\hat{u}$ from q_{initial} to q_{final} (Fig. 11(b)). For each $i = 1, \ldots, n$, we test if $q_i \in \mathcal{F}_\sigma$. If so, we add q_i as a root milestone in our roadmap. If not, we repeatedly sample other configurations in a growing neighborhood of q_i until we find some feasible $q_i' \in \mathcal{F}_\sigma$, which we add as a root instead of q_i.

Connecting neighboring roots with sampled trees. For $i = 1, \ldots, n-1$, we check if the root milestone q_i can be connected to its neighbor q_{i+1} with a feasible local path (as in [26]). If not, we add the pair of roots (q_i, q_{i+1}) to a list $\mathcal{R}$. Then, we apply PRM to grow trees between every pair in $\mathcal{R}$. For example, in Fig. 11(c) we add (q_2, q_3) and (q_4, q_5) to $\mathcal{R}$ and grow trees to connect both q_2 with q_3 and q_4 with q_5. We process all trees in parallel. So at every iteration, for each pair $(q_i, q_{i+1}) \in \mathcal{R}$, we first add m milestones to the trees at both q_i and q_{i+1} (in our experiments, we set $m = 5$). Then, we find the configurations q connected to q_i and q' connected to q_{i+1} that are closest. If q and q' can be connected by a local path, we remove (q_i, q_{i+1}) from $\mathcal{R}$. When we connect all neighboring roots, we return the resulting path; if this does not happen after a fixed number of iterations, we return failure. Just like our original implementation, this approach will find a path between q_{initial} and q_{final} whenever one exists (given enough time). However, since we seed our roadmap with milestones that are close to u, we expect the resulting motion to be similar (and of similar quality) to this primitive whenever possible (Fig. 11(d)), deviating significantly from it only when necessary.

4.2.2 Finding transitions

Again consider the robot at a configuration $q_{\text{initial}} \in \mathcal{F}_\sigma$ at a stance σ. But now, assume that we are only given a final stance σ', so we use a primitive u to guide our search for a transition before we plan a path to reach it.

Transforming the primitive to match σ and σ'. Since we do not know q_{final}, we can not use the same transformation (5) that we used for planning paths. Instead, we choose a rigid-body transformation of the form

$$\hat{u}(t) = Au(t) + b \tag{6}$$

that maps the nominal stances σ_u and σ'_u (associated with the primitive u) as closely as possible to the stances σ and σ'.

Recall that a stance consists of several contacts, each placing a link of the robot on the terrain. If we model the surface of the terrain and all robot links as a triangular mesh, then we can define the location of each placement by a finite number of points $r_i \in \mathbb{R}^3$. For example, the face-face contact between a foot and the ground might be defined by the vertices r_1, r_2, and r_3 of a triangle. We consider these points to be attached to the robot, so if the foot is placed against a different face in the terrain, the points r_1, r_2, and r_3 move in $\mathbb{R}^3$ but remain in the same location relative to the foot. We will use these points to define our mapping between stances.

In particular, let $r_i \in \mathbb{R}^3$ for $i = 1, \ldots, m$ be the set of all points defining the contacts in both σ_u and σ'_u, and let $s_i \in \mathbb{R}^3$ for $i = 1, \ldots, m$ be the set of all points defining the contacts in both σ and σ'. (We assume u has been chosen so that both sets have the same number of points.) Then we choose the rotation matrix A and translation b in (6) that minimize

$$\min_{A,b} \sum_i \|Ar_i + b - s_i\|_2^2.$$

We can compute A and b in closed form [3]. But, we only consider rotations A about the gravity vector to avoid tilting the robot into an unstable orientation.

Sampling a transition. As before, we sample configurations $q \in \mathcal{Q}_\sigma$ and keep them if $q \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$. But rather than sample configurations completely at random, we sample them in a growing neighborhood of $\hat{u}(1)$. We expect a well-chosen transition to further improve the quality of the path to reach it.

4.2.3 Finding contacts

Once more, consider the robot at a configuration $q_{\text{initial}} \in \mathcal{F}_\sigma$ and a stance σ. But now, assume we are given neither a final stance nor a transition, but only a primitive u. If u removes a robot link from the terrain, we immediately generate a final stance σ' by removing the corresponding contact from σ. But if u places a link in the terrain, we use it to guide our search for a new contact.

Transforming the primitive to match σ. We use the same transformation (6) to construct $\hat{u}$ as for finding transitions. But here, we compute A and b to map only σ_u to σ, since we do not know σ'. We use this transformation to adjust the placement of the new contact given by u. Let $r_i \in \mathbb{R}^3$ for $i = 1, \dots, m$ be the set of points defining this contact. Then the transformed contact is given by $\hat{r}_i = Ar_i + b$ for $i = 1, \dots, m$.

Sampling a contact. We define a sphere of radius δ, centered at $(1/m)\sum_i \hat{r}_i$. We increase δ until the intersection of this sphere with the terrain is non-empty (initially, we set δ approximately the size of either ATHLETE's or HRP-2's foot, respectively). We randomly sample a placement of the points $\hat{r}_i$ on the surface of the terrain inside the sphere, by first sampling a position of their centroid $s \in \mathbb{R}^3$ on the surface, then sampling a rotation of $\hat{r}_i$ about the surface normal at s. We check that the contact defined by this placement has similar properties (normal vector, friction coefficient) to the contact defined by u. If so, we add it to σ to form σ'. If not, we reject it and sample another placement.

4.2.4 Deciding which primitive to use

It only remains to decide which primitive u should be used, given an initial stance σ and configuration q_{initial}. We have experimented with a variety of heuristics. For example, we might pick the primitive that most closely matches σ_u with σ (in other words, that minimizes the error in a transformation of the form (6)). Likewise, we might pick the primitive that most closely matches σ'_u with the actual terrain. If no primitives match well, we use the basic method from Section 3 instead. However, the best approach is still not clear, and this issue remains an important area for future work.

4.3 Implementation and results

An example of climbing a single stair. With each additional part of a step that we compute using a primitive, we add to the quality of the result. For example, consider the motion of HRP-2 in Fig. 12 to climb a single stair of height 0.3m (just below the knee). This motion was planned from scratch, by randomly sampling contacts and transitions and by using PRM to generate paths. The robot does not look natural – its arm and leg motions are erratic, and its step over the stair is needlessly long. To improve this motion, we applied the two primitives shown in Fig. 10 (steps on flat ground). Fig. 13 shows the result of using these primitives to plan each path. Some erratic leg motions are eliminated, such as the backward movement of the leg in the second frame. The erratic arm motions remain, however, because the transition in the fourth frame is the same (still randomly sampled). Fig. 14 shows the result of using primitives to adjust this transition as well as to plan paths, eliminating most of the erratic arm motions. Finally, Fig. 15 shows the result of using primitives to select contacts well as plan transitions and paths. The chosen contact resulted in a much easier step, eliminating the extreme lean in the fifth frame.

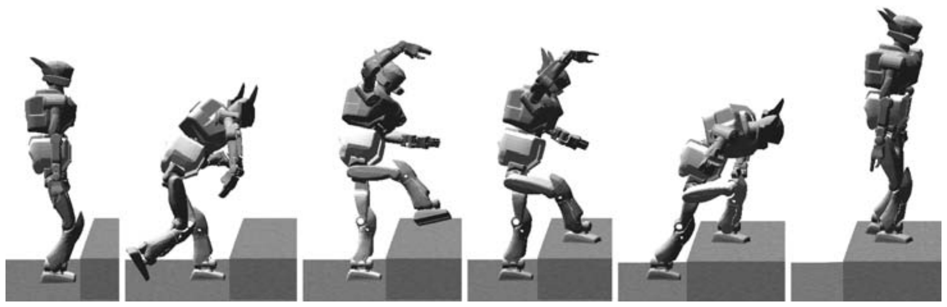


Figure 12: Stair step planned entirely from scratch.

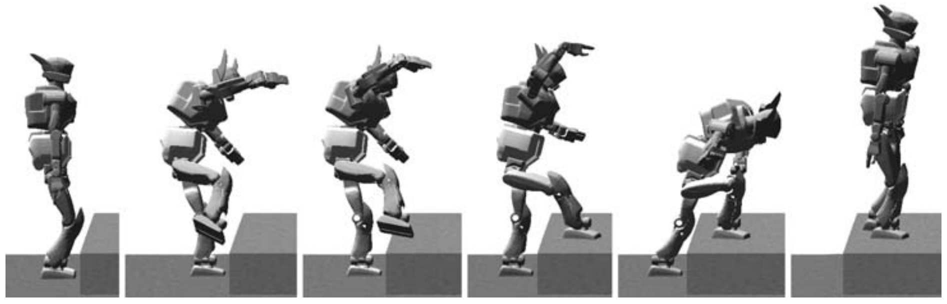


Figure 13: Primitives guide path planning, reducing unnecessary leg motions.

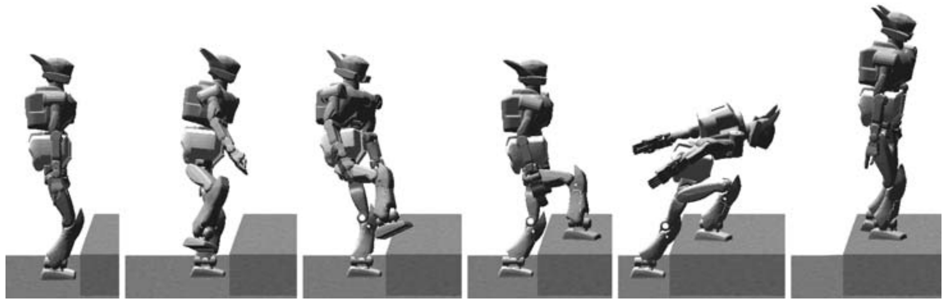


Figure 14: Primitives guide transition sampling, reducing unnecessary arm motions.

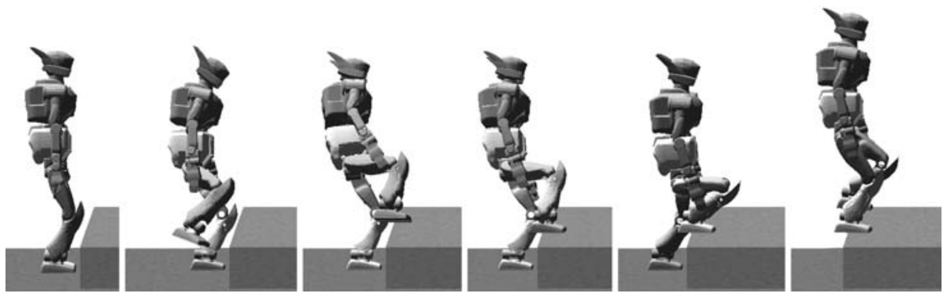
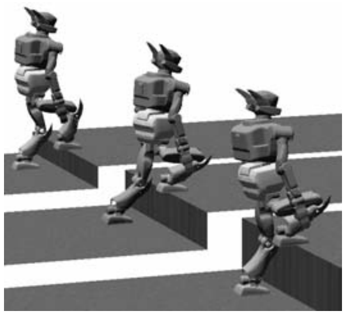


Figure 15: Primitives guide the choice of contact, resulting in an easier step.



Stair height	From scratch		Adapt primitive		Optimal objective
	Time	Objective	Time	Objective	
0.2m	8.61	5.03	5.42	3.04	2.19
0.3m	10.3	4.67	4.08	2.31	2.17
0.4m	12.2	5.15	10.8	3.27	2.55

Figure 16: Planning time and objective function values for stair steps, averaged over 5 runs.

Planning time and motion quality for stairs of different heights. In our experiments, we have observed that planning time remains low and motion quality remains high even when we use a primitive to plan a step that is quite different. For example, we adapted the same two primitives in Fig. 10 to stairs of height 0.2m, 0.3m, and 0.4m. Fig. 16 shows the results, averaged over five runs. Quality is measured by an objective function that penalizes both path length and deviations from an upright posture (lower values indicate higher quality). For comparison, we report the minimum objective value achieved after a lengthy off-line optimization. These results demonstrate that our use of primitives provides a modest reduction in planning time but significantly improves motion quality. Note also that both time and quality degrade gracefully as the step we are planning deviates further from the primitive.

Comparing motion primitives with gaits For ATHLETE, primitives work well in moderately difficult terrain, which is too difficult for gaits but not so irregular that the planner must explore all DOF to find a feasible path. Fig. 21 shows a fractal-generated terrain with moderate irregularities. A six-legged gait would lose stability and exceed torque limits at a number of locations (Fig. 21(a)). Using the gait as a motion primitive, the planner is able to find a natural-looking feasible path in about 8 minutes (Fig.21(b)).

More difficult terrains can only be traversed with a larger number of primitives. For example, on the problems in Table 1, the planner could not reliably adapt flat-ground gaits to stair steps higher than 0.2 times the diameter of the chassis, not much better than the gaits themselves.

A variety of other examples. We have tested our planner in many other example environments. Fig. 17 shows HRP-2 on uneven terrain (using the primitives in Fig. 10), in which the highest and lowest point differ by 0.5m. Fig. 18 shows HRP-2 climbing a ladder with rungs that have non-uniform spacing and that deviate from horizontal by up to 15°. The primitives for this example were generated on a ladder with horizontal, uniformly spaced rungs. Fig. 19 shows HRP-2 making several sideways steps among boulders, using the hands for support. Here, the primitives were generated by stepping sideways on flat ground while pushing against a vertical wall. Fig. 20 shows HRP-2 traversing very rough terrain with

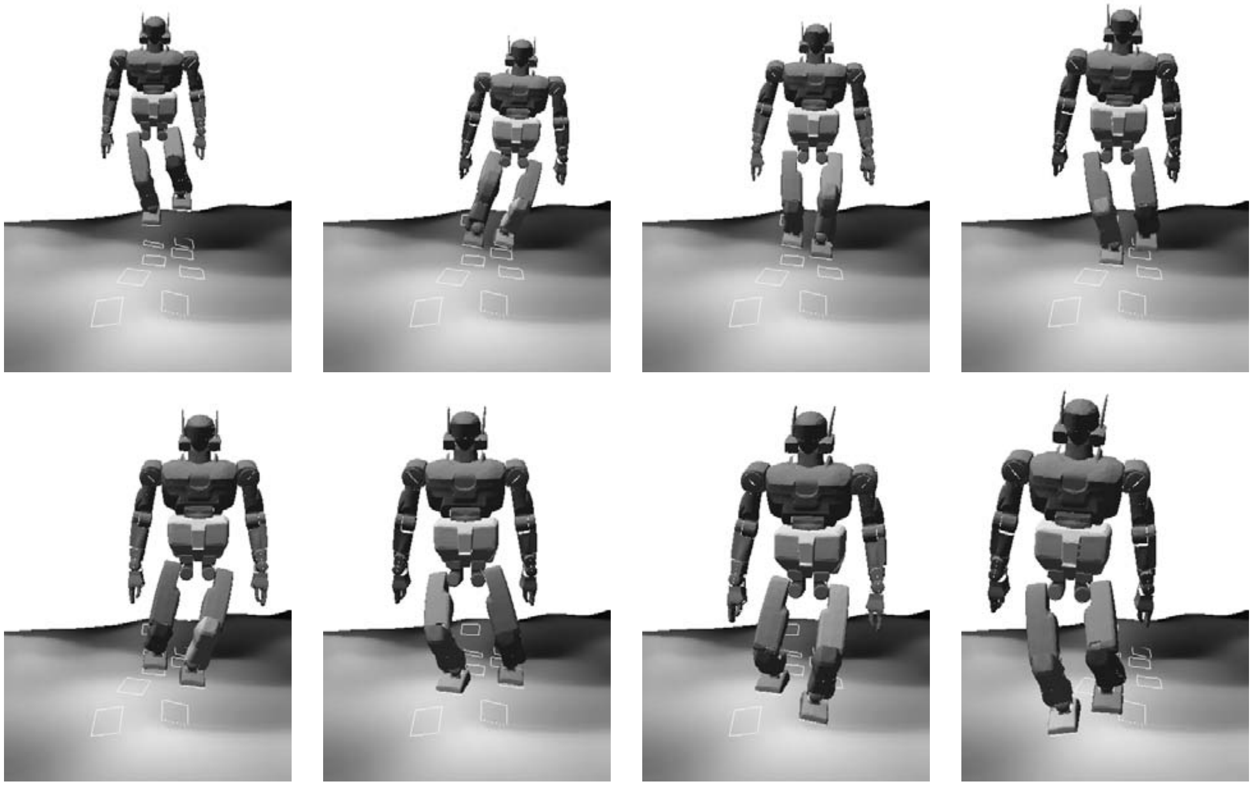


Figure 17: A planar walking primitive adapted to slightly uneven terrain.

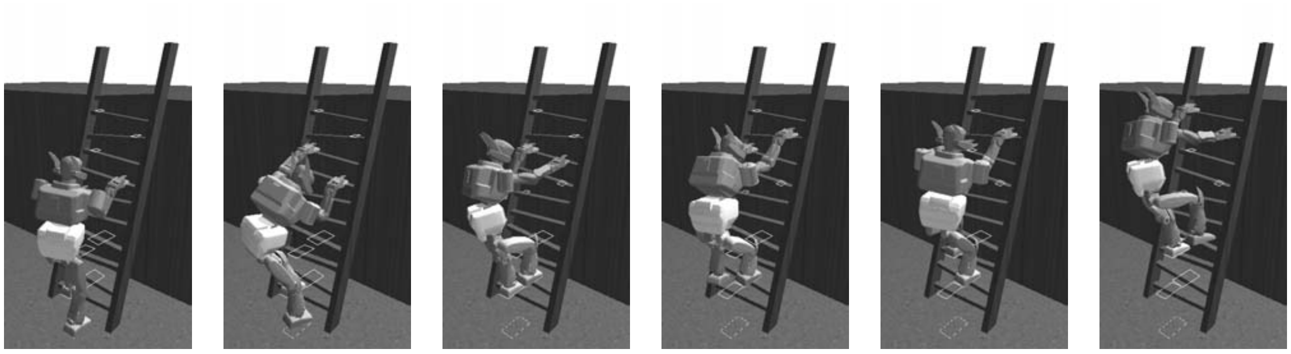


Figure 18: A ladder climbing primitive adapted to a new ladder with uneven rungs.

slopes up to 40°. This motion was generated with a larger set of primitives (including steps of several heights, a pivot step, and a high step using the hand for support). In all of these examples, contacts were sampled on-the-fly (using motion primitives), not placed by hand. Planning for the first three examples took about one minute on a 1.8 GHz PC. The fourth example took about eight minutes.

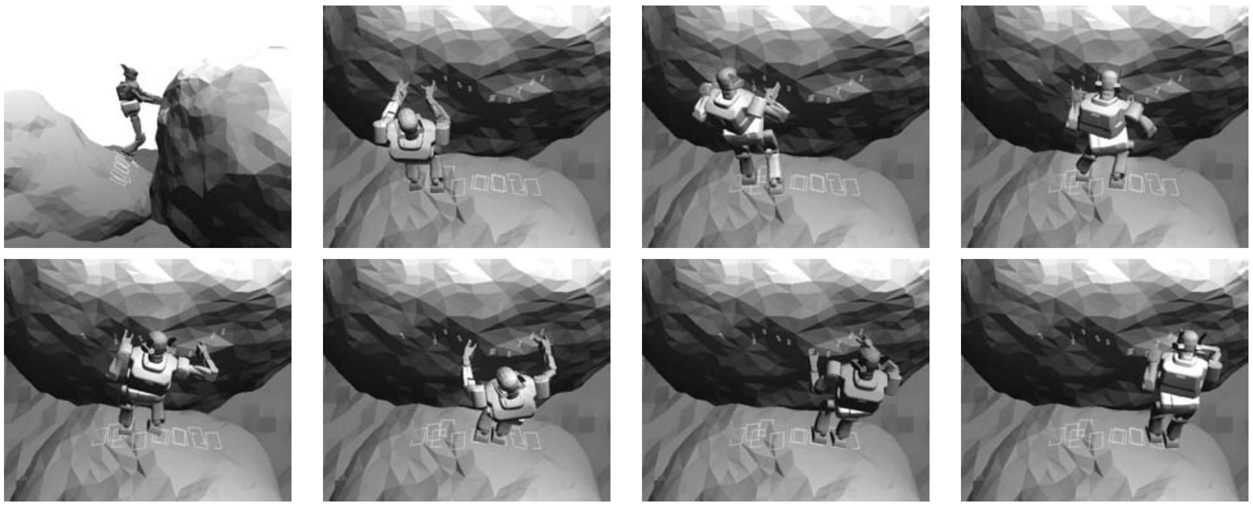


Figure 19: A side-step primitive using the hands for support, adapted to a terrain with large boulders. Hand support is necessary because the robot must walk on a highly sloped boulder.

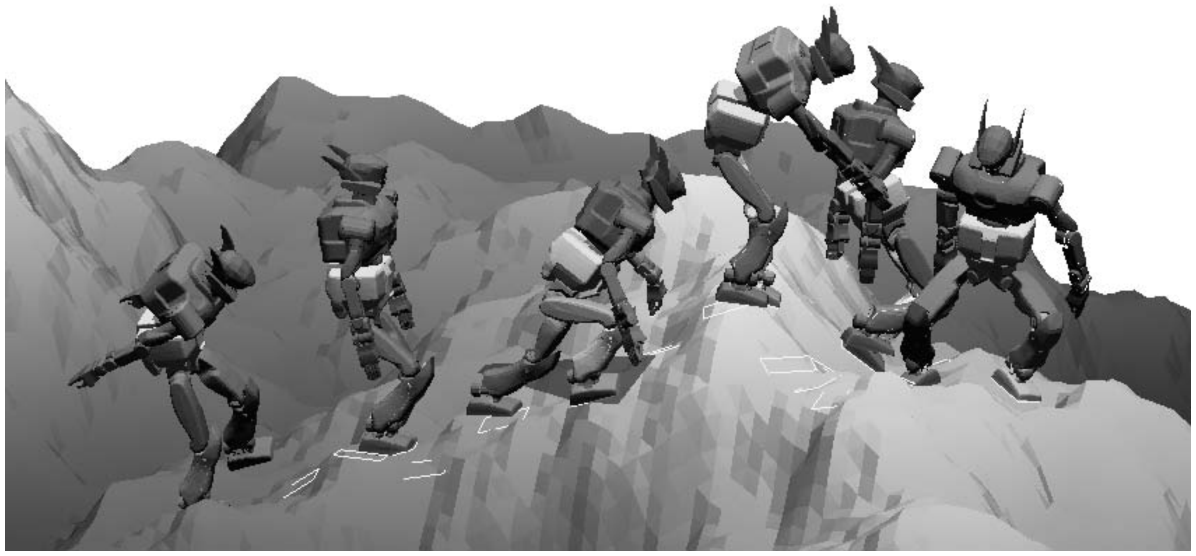


Figure 20: A motion on steep and uneven terrain generated from a set of several primitives. A hand is being used for support in the third configuration.

5 Conclusion

In this paper we described the design and implementation of a motion planner that enables legged robots with many degrees of freedom to walk safely across varied terrain. We focused on two robots in particular: the six-legged lunar vehicle ATHLETE(which has wheels on the end of each leg, but can fix these wheels to walk), and the humanoid HRP-2. Both robots are capable of walking carefully over terrain so rough that a fixed gait is insufficient. We

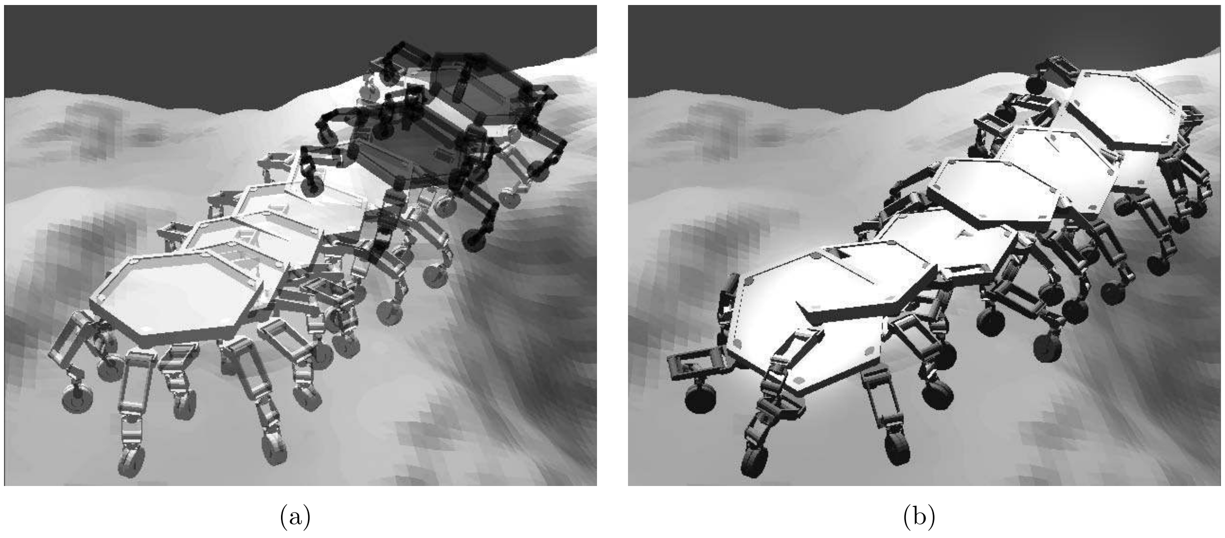


Figure 21: On moderately rough terrain, (a) using a gait fails (infeasible configurations are shaded), but (b) using this gait as a motion primitive succeeds.

made a key design choice in our planner – to choose footfalls before computing motions – because on this type of terrain, a legged robot's motion is most constrained just as it places or removes a foot. We presented several tools embedded in our planner (for sampling, local connection, and search heuristics) that extend previous techniques to satisfy the specific needs of ATHLETE and HRP-2. To improve motion quality, we also described how to derive a sampling strategy from a small set of motion primitives (such as a fixed gait on flat ground) that were generated offline. We demonstrated the flexibility of our planner with simulation results for both ATHLETE and HRP-2 that included both walking and rappelling motions on several example terrains.

There are many opportunities for future work. For example, our planner takes a reasonable amount of time for off-line computation (less than one hour), so it may help human tele-operators (for example, pilots at JPL) design difficult motions more quickly. A similar approach was used to plan motions for the recent Mars rovers. However, our planner is still too slow to be used on-the-fly (which may require computation times of less than five minutes). We are also developing better heuristics for deciding which motion primitives to generate and for choosing primitives appropriate to each step. Other important issues include incremental sensing and a consideration of dynamics, neither of which are addressed in this paper.

Acknowledgments

This work was partially supported by NSF grant 0412884 and by the RTLSM grant from NASA-JPL, specifically for the ATHLETE project. K. Hauser is supported by a Thomas V. Jones Stanford Graduate Fellowship.

References

[1] M. Akinc, K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki. Probabilistic roadmaps of trees for parallel computation of multiple query roadmaps. In *Int. Symp. Rob. Res.*, Siena, Italy, 2003.

[2] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Alg. Found. Rob.*, pages 109–125. A K Peters, Wellesley, MA, 1995.

[3] K. Arun, T. Huang, and S. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Machine Intell.*, 9(5):698–700, 1987.

[4] J. E. Bares and D. S. Wettergreen. Dante II: Technical description, results and lessons learned. *Int. J. Rob. Res.*, 18(7):621–649, 1999.

[5] D. Bevly, S. Farritor, and S. Dubowsky. Action module planning and its application to an experimental climbing robot. In *IEEE Int. Conf. Rob. Aut.*, pages 4009–4014, 2000.

[6] A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *IEEE Int. Conf. Rob. Aut.*, pages 348–353, San Francisco, CA, 2000.

[7] J. Bobrow, B. Martin, G. Sohl, E. Wang, F. Park, and J. Kim. Optimal robot motions for physical criteria. *J. of Robotic Systems*, 18(12):785–795, 2001.

[8] J.-D. Boissonnat, O. Devillers, and S. Lazard. Motion planning of legged robots. *SIAM J. Computing*, 30(1):218–246, 2000.

[9] T. Bretl. Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *Int. J. Rob. Res.*, 25(4):317–342, 2006.

[10] T. Bretl and S. Lall. A fast and adaptive test of static equilibrium for legged robots. In *IEEE Int. Conf. Rob. Aut.*, 2006.

[11] T. Bretl, J.-C. Latombe, and S. Rock. Toward autonomous free-climbing robots. In *Int. Symp. Rob. Res.*, Siena, Italy, 2003.

[12] R. Burridge, A. Rizzi, and D. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *Int. J. Rob. Res.*, 18(6):534–555, 1999.

[13] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. An active-set algorithm for nonlinear programming using linear programming and equality constrained subproblems. Technical Report OTC 2002/4, Optimization Technology Center, Northwestern University, Evanston, IL, 2002.

[14] H. Choset, K. Lynch, S. Hutchinson, G. Kanto, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, 2005.

[15] J. Cortés, T. Siméon, and J.-P. Laumond. A random loop generator for planning the motions of closed kinematic chains using prm methods. In *IEEE Int. Conf. Rob. Aut.*, Washington, D.C., 2002.

[16] C. Eldershaw and M. Yim. Motion planning of legged vehicles in an unstructured environment. In *IEEE Int. Conf. Rob. Aut.*, Seoul, South Korea, 2001.

[17] T. Estier, Y. Crausaz, B. Merminod, M. Lauria, R. Pguet, and R. Siegwart. An innovative space rover with extended climbing abilities. In *Space and Robotics*, Albuquerque, NM, 2000.

[18] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Trans. Robot.*, 25(1):116–129, 2002.

[19] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA J. of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[20] V. Gavrilets, E. Frazzoli, B. Mettler, M. Peidmonte, and E. Feron. Aggressive maneuvering of small autonomous helicopters: A human-centered approach. *Int. J. Rob. Res.*, 20(10):795–807, 2001.

[21] R. Geraerts and M. Overmars. Clearance based path optimization for motion planning. In *IEEE Int. Conf. Rob. Aut.*, New Orleans, LA, 2004.

[22] M. Gleicher. Retargetting motion to new characters. In *SIGGRAPH*, pages 33–42, 1998.

[23] S. Gottschalk, M. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Comput. Graph.*, 30:171–180, 1996.

[24] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.

[25] K. Harada, S. Kajita, H. Saito, F. Kanehiro, and H. Hirukawa. Integration of manipulation and locomotion by a humanoid robot. In *Int. Symp. Exp. Rob.*, Singapore, 2004.

[26] K. Hauser, T. Bretl, and J.-C. Latombe. Non-gaited humanoid locomotion planning. In *Humanoids*, Tsukuba, Japan, 2005.

[27] G. H. Heiken, D. T. Vaniman, and B. M. French. *Lunar Sourcebook: A User's Guide to the Moon*. Cambridge University Press, 1991.

[28] S. Hirose and O. Kunieda. Generalized standard foot trajectory for a quadruped walking vehicle. *Int. J. Rob. Res.*, 10(1):3–12, 1991.

[29] S. Hirose, K. Yoneda, and H. Tsukagoshi. Titan VII: Quadruped walking and manipulating robot on a steep slope. In *IEEE Int. Conf. Rob. Aut.*, pages 494–500, Albuquerque, NM, 1997.

[30] D. Hsu, J.-C. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. In *Int. Symp. Rob. Res.*, San Francisco, CA, 2005.

[31] K. Iagnemma, F. Genot, and S. Dubowsky. Rapid physics-based rough-terrain rover planning with sensor and control uncertainty. In *IEEE Int. Conf. Rob. Aut.*, Detroit, MI, 1999.

[32] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot HRP-2. In *IEEE Int. Conf. Rob. Aut.*, pages 1083–1090, New Orleans, LA, Apr. 2004.

[33] L. E. Kavraki, P. Svetska, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, 1996.

[34] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *IEEE Int. Conf. Rob. Aut.*, pages 945–952, San Diego, CA, 1994.

[35] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *SIGGRAPH*, pages 473–482, San Antonio, Texas, 2002.

[36] T. Kron and S. Y. Shin. Motion modeling for on-line locomotion synthesis. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 29–38, Los Angeles, CA, 2005.

[37] E. Krotkov and R. Simmons. Perception, planning, and control for autonomous walking with the ambler planetary rover. *Int. J. Rob. Res.*, 15:155–180, 1996.

[38] J. J. Kuffner, Jr. *Autonomous Agents for Real-Time Animation.* PhD thesis, Stanford University, 1999.

[39] J. J. Kuffner, Jr., K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots. In *Int. Symp. Rob. Res.*, Siena, Italy, 2003.

[40] J. Laumond, P. Jacobs, M. Taix, and R. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. Robot. Automat.*, 10(5):577–593, 1994.

[41] J.-P. Laumond. Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1120–1123, 1987.

[42] M. Lauria, Y. Piguet, and R. Siegwart. Octopus: an autonomous wheeled climbing robot. In *CLAWAR*, 2002.

[43] C. Lawrence, J. Zhou, and A. Tits. User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report TR-94-16r1, 20742, Institute for Systems Research, University of Maryland, College Park, MD, 1997.

[44] H. Lee, Y. Shen, C.-H. Yu, G. Singh, and A. Y. Ng. Quadruped robot obstacle negotiation via reinforcement learning. In *IEEE Int. Conf. Rob. Aut.*, 2006.

[45] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition. In *Advances in Neural Information Processing Systems*, 2005.

[46] C. K. Liu, A. Hertzmann, and Z. Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph.*, 24(3):1071–1081, 2005.

[47] M. Meredith and S. Maddock. Adapting motion capture data using weighted real-time inverse kinematics. *Comput. Entertain.*, 3(1), 2005.

[48] P. E. Missiuro and N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. To appear in IEEE Int. Conf. Rob. Aut., 2006.

[49] E. Mumm, S. Farritor, P. Pirjanian, C. Leger, and P. Schenker. Planetary cliff descent using cooperative robots. *Autonomous Robots*, 16:259–272, 2004.

[50] A. Y. Ng, H. J. Kim, M. Jordan, and S. Sastry. Autonomous helicopter flight via reinforcement learning. In *Neural Information Processing Systems 16*, 2004.

[51] C. L. Nielsen and L. E. Kavraki. A two level fuzzy prm for manipulation planning. In *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, pages 1716–1721, Takamatsu, Japan, 2000.

[52] A. Okamura, N. Smaby, and M. Cutkosky. An overview of dexterous manipulation. In *IEEE Int. Conf. Rob. Aut.*, pages 255–262, 2000.

[53] D. K. Pai, R. A. Barman, and S. K. Ralph. Platonic beasts: Spherically symmetric multilimbed robots. *Autonomous Robots*, 2(4):191–201, 1995.

[54] J. Pettré, J.-P. Laumond, and T. Siméon. A 2-stages locomotion planner for digital actors. In *Eurographics/SIGGRAPH Symp. Comp. Anim.*, 2003.

[55] M. B. Popovic, A. Goswami, and H. Herr. Ground reference points in legged locomotion: Definitions, biological trajectories and control implications. *Int. J. Rob. Res.*, 24(12):1013–1032, 2005.

[56] Z. Popović and A. Witkin. Physically based motion transformation. In *SIGGRAPH*, pages 11–20, 1999.

[57] L. Ren, A. Patrick, A. A. Efros, J. K. Hodgins, and J. M. Rehg. A data-driven approach to quantifying natural human motion. *ACM Trans. Graph.*, 24(3):1090–1097, 2005.

[58] A. Sahbani, J. Cortés, and T. Siméon. A probabilistic algorithm for manipulation planning under continuous grasps and placements. In *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, pages 1560–1565, Lausanne, Switzerland, 2002.

[59] G. Sánchez and J.-C. Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *Int. J. of Rob. Res.*, 21(1):5–26, 2002.

[60] F. Schwarzer, M. Saha, and J.-C. Latombe. Exact collision checking of robot paths. In *WAFR*, Nice, France, Dec 2002.

[61] L. Sentis and O. Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *Int. J. Humanoid Robotics*, 2(4):505–518, 2005.

[62] A. Shapiro and E. Rimon. PCG: A foothold selection algorithm for spider robot locomotion in 2d tunnels. In *IEEE Int. Conf. Rob. Aut.*, pages 2966–2972, Taipei, Taiwan, 2003.

[63] H. J. Shin, J. Lee, S. Y. Shin, and M. Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. Graph.*, 20(2):67–94, 2001.

[64] G. Song, S. Miller, and N. M. Amato. Customizing PRM roadmaps at query time. In *IEEE Int. Conf. Rob. Aut.*, pages 1500–1505, Seoul, Korea, 2001.

[65] S.-M. Song and K. J. Waldron. *Machines that walk: the adaptive suspension vehicle.* The MIT Press, Cambridge, MA, 1989.

[66] M. Stilman and J. J. Kuffner. Planning among movable obstacles with artificial constraints. In *WAFR*, New York, NY, 2006.

[67] S. G. Vougioukas. Optimization of robot paths computed by randomized planners. In *IEEE Int. Conf. Rob. Aut.*, Barcelona, Spain, 2005.

[68] L. Wang and C. Chen. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Trans. Robot. Automat.*, 7(4):489–499, 1991.

[69] D. Wettergreen, C. Thorpe, and W. Whittaker. Exploring mount erebus by walking robot. *Robotics and Autonomous Systems*, 11:171–185, 1993.

[70] B. H. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmad, A. Sirota, and B. Cooper. ATHLETE: a cargo handling and manipulation robot for the moon. *Journal of Field Robotics*, 24(5):421–434, 2007.

[71] A. Witkin and Z. Popović. Motion warping. In *SIGGRAPH*, pages 105–108, Los Angeles, CA, 1995.

[72] J. H. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Trans. Robot. Automat.*, 17(6):951–958, 2001.

[73] K. Yamane, J. J. Kuffner, and J. K. Hodgins. Synthesizing animations of human manipulation tasks. *ACM Trans. Graph.*, 23(3):532–539, 2004.

[74] K. Yoneda, F. Ito, Y. Ota, and S. Hirose. Steep slope locomotion and manipulation mechanism with minimum degrees of freedom. In *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, pages 1897–1901, 1999.

[75] Y. F. Zheng and J. Shen. Gait synthesis for the SD-2 biped robot to climb sloping surface. *IEEE Trans. Robot. Automat.*, 6(1):86–96, 1990.