Motion planning for a six-legged lunar robot

Kris Hauser[1], Timothy Bretl[1], Jean-Claude Latombe[1], and Brian Wilcox[2]

[1] Computer Science Department, Stanford University
{khauser,tbretl}@stanford.edu
latombe@cs.stanford.edu
[2] Jet Propulsion Laboratory, California Institute of Technology
Brian.H.Wilcox@jpl.nasa.gov

Summary. This paper studies the motion of a large and highly mobile six-legged lunar vehicle called ATHLETE, developed by the Jet Propulsion Laboratory. This vehicle rolls on wheels when possible, but can use the wheels as feet to walk when necessary. While gaited walking may suffice for most situations, rough and steep terrain requires novel sequences of footsteps and postural adjustments that are specifically adapted to local geometric and physical properties. This paper presents a planner to compute these motions that combines graph searching techniques to generate a sequence of candidate footfalls with probabilistic sample-based planning to generate continuous motions to reach them. The viability of this approach is demonstrated in simulation on several example terrains, even one that requires rappelling.

1 Introduction

In this paper we describe the design and implementation of a motion planner for a six-legged lunar vehicle called ATHLETE (All-Terrain Hex-Limbed Extra-Terrestrial Explorer), shown in Fig 1. This large and highly mobile vehicle was developed by the Jet Propulsion Laboratory (JPL).[3] It can roll rapidly on rotating wheels over flat smooth terrain and walk carefully on fixed wheels over irregular and steep terrain. In particular, 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 [19].

On this type of terrain, ATHLETE's walking motion is governed largely by two interdependent constraints: *contact* (keep wheels, or *feet*, at a carefully

[3] The views presented in this paper do not reflect those of NASA or JPL.

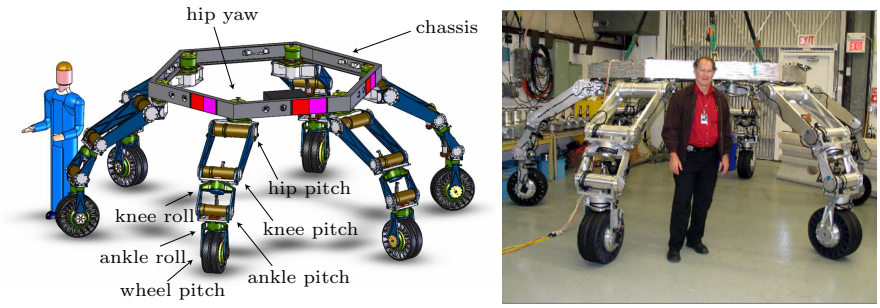


Fig. 1. The ATHLETE lunar vehicle (developed by JPL).

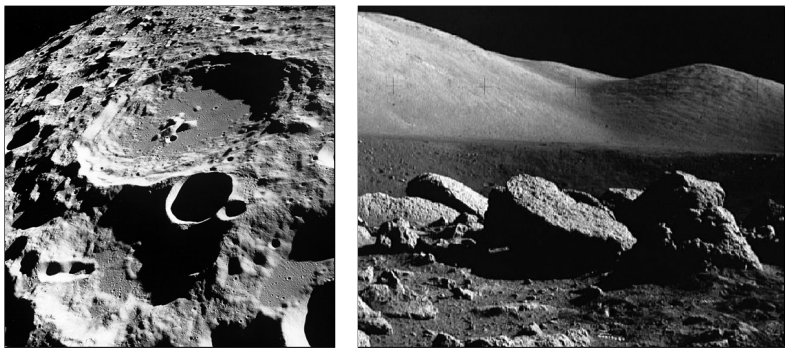


Fig. 2. Pictures of lunar terrain from Apollo missions [19].

chosen set of footfalls) and *equilibrium* (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 (for example, average slope) and their physical properties (for example, coefficient of friction), both of which vary across the terrain. So every time ATHLETE takes a step, it faces a dilemma: it can't know the constraints on its subsequent motion until it chooses a footfall, a choice it can't make until it knows where it will step next. Direct teleoperation does not help to resolve this dilemma – on the contrary, teleoperation can be difficult and painfully slow for robots like ATHLETE [6].

To handle this dilemma in our planner, we make a key design choice (Section 3) – *to choose footfalls before computing motions.* We begin by identifying a number of potentially useful footfalls across the terrain. Each mapping of ATHLETE'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). ATHLETE 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 refine this sequence into a feasible, continuous trajectory by finding paths between subsequent transitions. We do this because ATHLETE's motion on irregular and steep terrain is most constrained just as it places a foot at or removes a foot from a footfall. At this instant, ATHLETE must be able to reach the footfall (contact) but can not use it to avoid falling (equilibrium). So footfalls are the "bottleneck" 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.

We implement our planner using an approach similar to [6] and [18] that combines graph searching techniques to generate a sequence of candidate footfalls with probabilistic sample-based planning to generate continuous motions to reach them. But several key tools embedded in this framework (Section 4) are tailored specifically to ATHLETE. We need a method of sampling feasible configurations (from scratch as well as via perturbation) and of connecting pairs of configurations with local paths, hard since ATHLETE has many degrees of freedom and many closed-loop chains. We also need a heuristic to generate footfalls and to guide our search through the collection of stances, hard since lunar terrain is difficult (so careful selection of footfalls is important) but not extreme (so the number of candidate stances is enormous). Finally, we need to smooth ATHLETE's motion both to look natural when interacting with a human operator – hard since the robot is not anthropomorphic – and to help avoid disturbing the ground (for example, by toppling rock).

Simulation results (Section 5) demonstrate the viability of our approach. We also show the flexibility of our implementation by adapting it to rappelling as well as walking motions of ATHLETE.

2 Related work

2.1 Application

Some humanoids are capable of walking over somewhat uneven terrain [28,49]. Other legged robots are capable of walking over rougher terrain, including quadrupeds [20], hexapods [26,43], parallel walkers [48], and spherically symmetric robots [35]. Wheeled robots with active or rocker-bogie suspension can also traverse rough terrain by changing wheel angles and center of mass position [14,23,29]. Careful descent is possible by rappelling as well, using either legs [3,21,46] or wheels [32]. The terrain we consider for ATHLETE is even more irregular and steep than in most previous applications, although not as steep as for free-climbing robots [6].

Careful walking also resembles dexterous manipulation. ATHLETE grasps the terrain like a hand grasps an object, placing and removing footfalls rather than finger contacts. ATHLETE has to remain in equilibrium as it moves (only the object must remain in equilibrium during manipulation), and uses fewer

contact modes while walking (no sliding or rolling), but still faces similar challenges [4, 34]. 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 ATHLETE takes a sequence of steps [2].

2.2 Planning

In order to walk, ATHLETE must plan both a sequence of footfalls and continuous motions to reach them. Previous approaches differ primarily in which part of the problem they consider first:

(a) *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 [25]. Similarly, a humanoid planner might generate a 2-D collision-free path of a bounding cylinder, then follow this path with a fixed gait [27, 36]. A related strategy is to plan a path for the center of mass, then to compute footfalls and limb motions that keep the center of mass stable [13]. These techniques are fast, but do not extend well to irregular and steep terrain.

(b) *Footfalls before motion.* When the choice of contact location is critical, it makes sense to compute a sequence of footfalls first. Most work is based on the approach to manipulation planning proposed by [2], which expresses connectivity between different states of contact as a graph. For "spider-robots" walking on horizontal terrain, the exact structure of this graph can be computed quickly using analytical techniques [5]. For more general systems, the graph can sometimes be simplified by assuming partial gaits, for example restricting the order in which limbs are moved [40] or restricting footsteps to a discrete set [28]. But when motion is distinctly non-gaited (as in manipulation planning [33, 37], free-climbing [6], or for ATHLETE), each step requires the exploration of configuration space. This motivates the two-stage search strategy we adopt in Section 3.

2.3 Key tools

Each of the tools embedded in our planner improves and extends previous techniques to satisfy the specific needs of ATHLETE:

(a) *Sampling and local connection.* We use a variant of the Probabilistic-Roadmap (PRM) approach (see Chap. 7 of [11]) to generate transitions between stances (configurations that are feasible at both one stance and another) as well as paths between transitions. A PRM planner samples configurations at random, retaining feasible ones as milestones and connecting close milestones if possible with feasible local paths. Its performance depends on fast methods

of sampling and local connection, either from scratch across all of configuration space [24] or via perturbation by growing trees from existing milestones [1, 22, 30]. Closed kinematic chains (ATHLETE has many) make both of these operations harder because there is zero probability that an arbitrary configuration will satisfy the closure constraints. One approach breaks chains into "active" and "passive" joints, sampling a configuration of the active joints and using analytical inverse kinematics to solve for the rest [12, 17]. Another approach uses numerical optimization to move a configuration onto the constraint manifold [18, 45, 47]. We use a combination of these two methods.

(b) *Heuristics for footfall selection.* A variety of heuristics have been proposed for estimating the usefulness of a footfall. Most are geometric criteria that determine how flat a footfall is [9, 10, 31]. On irregular and steep terrain, however, the usefulness of a footfall also depends on its location with respect to other footfalls – in particular, on how these footfalls are combined in each stance. We use these heuristics to guide the search for a candidate sequence of stances to reach a goal position, similar to [10, 41].

(c) *Path smoothing.* Paths generated by a PRM planner are feasible, but not necessarily optimal. A number of methods have been suggested to improve the result, including "short-cut" heuristics [24, 42] and gradient descent algorithms [15, 44]. We use a similar approach. But in addition to being safe and efficient, ATHLETE's motions must also "look good" to human operators.

3 Design of the motion planner

3.1 Motion constraints

A *configuration* of ATHLETE, denoted q, is a parameterization of the robot's placement in 3-D space. In the following, 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). The set of all such q is the *configuration space*, denoted $\mathcal{Q}$, of dimensionality 42.

When ATHLETE is walking, a brake is applied to each wheel so it can not roll. In this case, we call each wheel a *foot*. Whenever a foot is placed in contact with the terrain, we call this placement (the fixed position and orientation of a wheel in 3-D space) a *footfall*. Since all feet are identical, potentially any foot could be placed at any footfall. We call a specific mapping of feet to footfalls a *stance*. Consider a stance σ with $3 \leq N \leq 6$ footfalls (in general, at least three are required to achieve statically stable equilibrium). The feasible space $\mathcal{F}_\sigma$ is the set of all feasible configurations of the robot at stance σ. To be in $\mathcal{F}_\sigma$, a configuration q must satisfy several constraints:

(a) *Contact.* The N legs whose feet are in contact with the ground 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 $42 - 6N$,

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.

(b) *Static equilibrium.* To remain balanced, ATHLETE must be able to apply forces with its feet on the terrain that compensate for gravity without slipping. A necessary condition is that ATHLETE's center of mass (CM) lie above a *support polygon*. But on irregular and steep terrain, the support polygon does not always correspond to the base of ATHLETE's feet. For example, ATHLETE will slip off a flat and featureless slope that is too steep, regardless of its CM position. To compute the support polygon, we model the contact interface at each footfall 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 ATHLETE's CM (which varies with its configuration). Assume ATHLETE 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 ATHLETE 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 [6–8]. 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 [7].

(c) *Joint torque limits.* The above equilibrium test assumes ATHLETE 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, and let $\|\cdot\|$ be a weighted L_∞ norm where $\|\tau\| < 1$ implies that each joint torque is within bounds. Then

we check joint torque limits by computing τ that achieves equilibrium with minimum $\|\tau\|$ (a linear program), and verify $\|\tau\| < 1$.

(d) *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 [16, 39].

3.2 Two-stage search

To walk from once place to another, ATHLETE has to take a sequence of steps. Formally, we define a *step* as any continuous motion at a fixed stance that terminates by either placing or removing a foot. In particular, let σ and σ' be the stances before and after a step, respectively. Then this step is a continuous path from the robot's current configuration $q_{\text{initial}} \in \mathcal{F}_\sigma$ to some configuration $q_{\text{final}} \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ that we call a *transition*. During this step, ATHLETE may move all legs simultaneously, but we assume that no two feet are placed or removed simultaneously. Therefore, σ and σ' differ only by a single footfall, which is present in only one of the two stances.

We encode the connectivity among stances 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 ATHLETE to take a step from one stance to another. Both necessary and sufficient conditions are provided by 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 $\mathcal{F}_\sigma$. The stance and transition graphs represent the connectivity of ATHLETE'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 [6]. The algorithm EXPLORE-STANCEGRAPH searches the stance graph (Fig. 3). 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. 3). 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, ATHLETE's motion on irregular and steep terrain is most constrained just as it places or removes a

```
EXPLORE-STANCEGRAPH(q_initial, σ_initial, σ_final)
 1  Q ← {σ_initial}
 2  while Q is nonempty do
 3      unstack a node σ from Q
 4      if σ = σ_final then
 5          construct a path [σ_1, ..., σ_n] from σ_initial to σ_final
 6          i ← EXPLORE-TRANSITIONGRAPH(σ_1, ..., σ_n, q_initial)
 7          if i = n then
 8              return the multi-step motion
 9          else
10              delete the edge (σ_i, σ_{i+1}) from the stance graph
11      else
12          for each unexplored stance σ' adjacent to σ do
13              if FIND-TRANSITION(σ, σ') then
14                  add a node σ' and an edge (σ, σ')
15                  stack σ' in Q
16  return "failure"
EXPLORE-TRANSITIONGRAPH(σ_i, ..., σ_n, q)
1  i_max ← i
2  for q' ← FIND-TRANSITION(σ_i, σ_{i+1}) in each component of F_{σ_i} ∩ F_{σ_{i+1}} do
3      if FIND-PATH(σ_i, q, q') then
4          i_cur ← EXPLORE-TRANSITIONGRAPH(σ_{i+1}, ..., σ_n, q')
5          if i_cur = n then
6              return n
7          elseif i_cur > i_max then
8              i_max = i_cur
9  return i_max
```

Fig. 3. Algorithms to explore the stance graph and the transition graph.

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 $\mathcal{F}_\sigma$.

A number of tools are embedded in this framework (the subroutines FIND-TRANSITION and FIND-PATH, a heuristic for ordering Q, and a method of smoothing the resulting motion) that we discuss in the following section.

4 Tools to support the motion planner

4.1 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 [12, 45, 47], 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 [18], we do this in two steps:

(a) *Create a candidate configuration that is close to $\mathcal{Q}_\sigma$.* 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 either reach or come closest to reaching each 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.

(b) *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. We represent the error in position and orientation of each foot 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 it 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.

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

```
FREE-PATH(q, q')
 1  if the distance from q to q' is less than ε then
 2      return TRUE
 3  q_mid ← (q + q')/2
 4  if Newton-Raphson from q_mid results in q_mid ∈ Q_σ then
 5      if q_mid ∈ F_σ then
 6          return (FREE-PATH(q, q_mid) & FREE-PATH(q_mid, q'))
 7      else
 8          return FALSE
 9  else
10      return FALSE
```

Fig. 4. Algorithm to connect close configurations with a local path.

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.

4.2 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 bidirectional (growing trees from both q and q') and lazy (delaying the creation of local paths until a candidate sequence of milestones is found) [38].

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 [47]).

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. 4). At each iteration, FREE-PATH first applies Newton-Raphson (see Section 4.1) 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'.

4.3 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 [33].
- *Distance to goal.* We increase the cost of σ proportional to the distance between the centroid of its footfalls 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 footfalls and those of a nominal stance on flat ground (with footfalls 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'.

4.4 Path smoothing

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. To improve the result, we apply a method of smoothing similar to [15,44], which uses gradient descent to achieve criteria like minimum path length and maximum clearance (or safety margin). However, we modify this approach in two ways. First, ATHLETE's motion consists of a sequence of short paths (steps) through separate feasible spaces rather than a single path through one feasible space. We consider this entire sequence of paths at once (deforming transitions as well as paths) rather than each one individually. So during the optimization, different parts of ATHLETE's motion are subject to different constraints. Second, because ATHLETE is expected to interact with humans, we try to make its motion "look good" to human operators. We do this by allowing the operator to select, ahead of time, a small set of nominal configurations (for example, standing on six legs, standing on three legs, or crouching). Then, in addition to minimizing path length and maximizing clearance, we also minimize deviation from any point q along the path to the closest nominal configuration q'. Even a small number of iterations (taking about 10 minutes on a 2GHz PC) makes a noticeable difference in motion quality.

5 Implementation and results

We tested our planner in simulation on several example terrains. Each terrain is a height-map of the form $z = f(x, y)$, created using a fractal generation method and represented by a triangular mesh consisting of 32768 triangles, each about the size of one of ATHLETE's wheels. Currently, we randomly sample 200 footfalls in each terrain to use in our planner, relying on our graph search heuristic (Section 4.3) to 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.

First, we show that our planner enables ATHLETE to walk across varied terrain. Fig. 5 shows motion on smooth, undulating ground (where all contacts are modeled with the same coefficient of friction). The initial and final stances are at a distance of about twice the radius of ATHLETE's chassis. The resulting motion consisted of 66 steps. Total computation time was 14 minutes. Fig. 6 shows motion on irregular and steep ground. The resulting motion consisted of 84 steps. Total computation time was 26 minutes. For comparison, Fig. 7 shows the result of applying a common fixed gait (an alternating-tripod) to both of these terrains. On smooth ground, the gait works well – it is simpler to plan, and results in more efficient motion (Fig. 7(a)). On irregular and steep ground, however, the gait does not work at all – it causes ATHLETE to lose balance or exceed torque limits at several locations (Fig. 7(b)).

Our results also demonstrate that the planner is flexible enough to handle different robot morphologies. Fig. 8 shows 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.

6 Conclusion

In this paper we described the design and implementation of a motion planner for a six-legged lunar vehicle called ATHLETE, developed by JPL. This vehicle has wheels on the end of each leg, but can fix these wheels to walk carefully over terrain so rough that a fixed gait is insufficient. We made a key design choice in our planner – to choose footfalls before computing motions – because on this type of terrain, ATHLETE'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, search heuristics, and path smoothing) that extend previous techniques to satisfy the specific needs of ATHLETE. We demonstrated the flexibility of our planner with simulation results 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

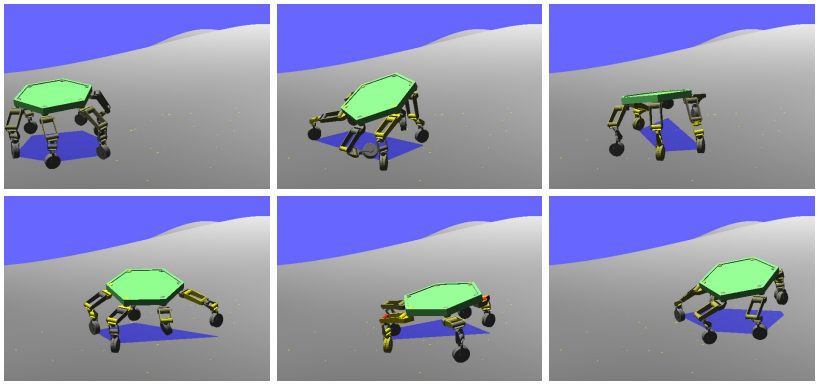


Fig. 5. Walking on smooth, undulating terrain with no fixed gait.

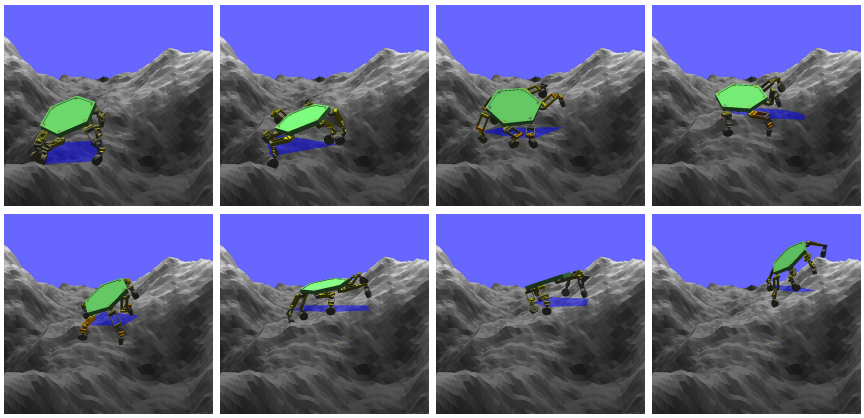
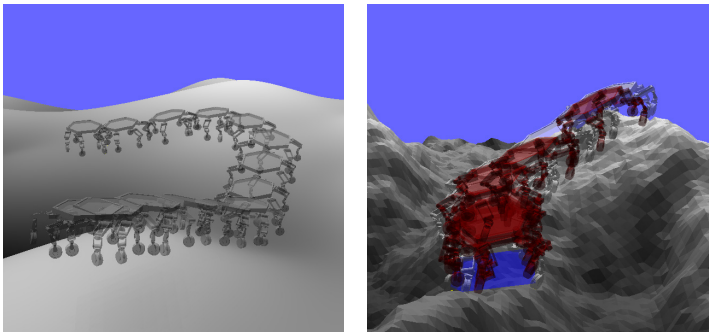


Fig. 6. Walking on steep, uneven terrain with no fixed gait.



(a) (b)

Fig. 7. Walking with an alternating tripod gait is (a) feasible on smooth terrain but (b) infeasible on uneven terrain. Infeasible configurations are highlighted red.

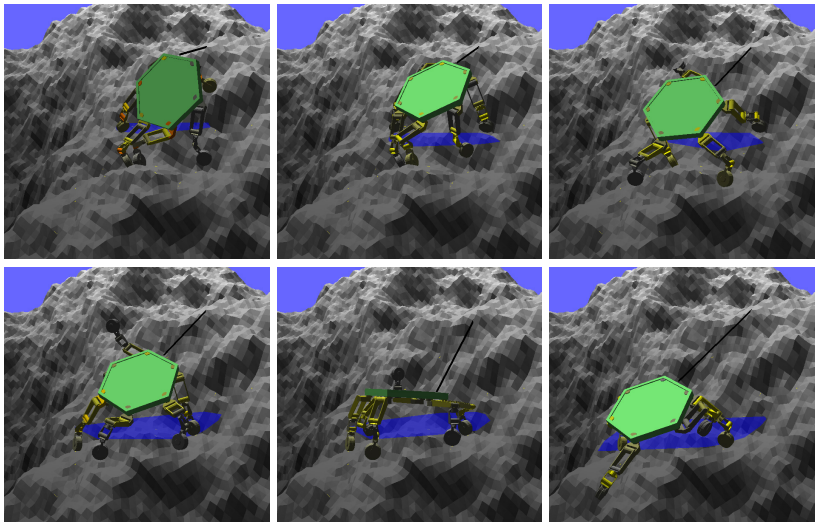


Fig. 8. Rappelling down an irregular 60° slope with no fixed gait.

hour), so it may help human 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 working to derive motion strategies or other methods of model reduction to address this problem. Other important issues include incremental sensing and a consideration of dynamics.

Acknowledgments. This work was supported by the RTLSM grant from NASA-JPL, specifically for the ATHLETE project.

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. J. E. Bares and D. S. Wettergreen. Dante II: Technical description, results and lessons learned. *Int. J. Rob. Res.*, 18(7):621–649, 1999.
4. A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *IEEE Int. Conf. Rob. Aut.*, pages 348–353, San Francisco, CA, 2000.
5. J.-D. Boissonnat, O. Devillers, and S. Lazard. Motion planning of legged robots. *SIAM J. Computing*, 30(1):218–246, 2000.
6. T. Bretl. Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. To appear in *Int. J. Rob. Res.*, 2006.
7. T. Bretl and S. Lall. A fast and adaptive test of static equilibrium for legged robots. To appear in IEEE Int. Conf. Rob. Aut., 2006.

8. T. Bretl, J.-C. Latombe, and S. Rock. Toward autonomous free-climbing robots. In *Int. Symp. Rob. Res.*, Siena, Italy, 2003.
9. C. Caillas, M. Hebert, E. Krotkov, I. Kweon, and T. Kanade. Methods for identifying footfall positions for a legged robot. In *Int. Work. Int. Rob. Sys.*, pages 244–250, 1989.
10. J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami. Planning biped navigation strategies in complex environments. In *IEEE Int. Conf. Hum. Rob.*, Munich, Germany, 2003.
11. 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.
12. 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.
13. C. Eldershaw and M. Yim. Motion planning of legged vehicles in an unstructured environment. In *IEEE Int. Conf. Rob. Aut.*, Seoul, South Korea, 2001.
14. 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.
15. R. Geraerts and M. Overmars. Clearance based path optimization for motion planning. In *IEEE Int. Conf. Rob. Aut.*, New Orleans, LA, 2004.
16. S. Gottschalk, M. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. In *ACM SIGGRAPH*, pages 171–180, 1996.
17. L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *WAFR*, 2000.
18. K. Hauser, T. Bretl, and J.-C. Latombe. Non-gaited humanoid locomotion planning. In *Humanoids*, Tsukuba, Japan, 2005.
19. G. H. Heiken, D. T. Vaniman, and B. M. French. *Lunar Sourcebook: A User's Guide to the Moon*. Cambridge University Press, 1991.
20. S. Hirose and O. Kunieda. Generalized standard foot trajectory for a quadruped walking vehicle. *Int. J. Rob. Res.*, 10(1):3–12, 1991.
21. 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.
22. D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *IEEE Int. Conf. Rob. Aut.*, pages 2219–2226, 1997.
23. 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.
24. 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.
25. Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *IEEE Int. Conf. Rob. Aut.*, pages 945–952, San Diego, CA, 1994.
26. 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.
27. J. J. Kuffner, Jr. *Autonomous Agents for Real-Time Animation*. PhD thesis, Stanford University, 1999.
28. 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.

29. M. Lauria, Y. Piguet, and R. Siegwart. Octopus: an autonomous wheeled climbing robot. In *CLAWAR*, 2002.
30. S. M. LaValle and J. J. Kuffner, Jr. Randomized kinodynamic planning. *Int. J. Rob. Res.*, 20(5):379–400, 2001.
31. K. Low and S. Bai. Terrain-evaluation-based motion planning for legged locomotion on irregular terrain. *Adv. Rob.*, 17(8):761–778, 2003.
32. E. Mumm, S. Farritor, P. Pirjanian, C. Leger, and P. Schenker. Planetary cliff descent using cooperative robots. *Autonomous Robots*, 16:259–272, 2004.
33. 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.
34. A. Okamura, N. Smaby, and M. Cutkosky. An overview of dexterous manipulation. In *IEEE Int. Conf. Rob. Aut.*, pages 255–262, 2000.
35. D. K. Pai, R. A. Barman, and S. K. Ralph. Platonic beasts: Spherically symmetric multilimbed robots. *Autonomous Robots*, 2(4):191–201, 1995.
36. J. Pettré, J.-P. Laumond, and T. Siméon. A 2-stages locomotion planner for digital actors. In *Eurographics/SIGGRAPH Symp. Comp. Anim.*, 2003.
37. 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.
38. 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.
39. F. Schwarzer, M. Saha, and J.-C. Latombe. Exact collision checking of robot paths. In *WAFR*, Nice, France, Dec 2002.
40. 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.
41. S. Singh, R. Simmons, T. Smith, A. T. Stentz, V. Verma, A. Yahja, and K. Schwehr. Recent progress in local and global traversability for planetary rovers. In *IEEE Int. Conf. Rob. Aut.*, 2000.
42. 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.
43. S.-M. Song and K. J. Waldron. *Machines that walk: the adaptive suspension vehicle*. The MIT Press, Cambridge, MA, 1989.
44. S. G. Vougioukas. Optimization of robot paths computed by randomized planners. In *IEEE Int. Conf. Rob. Aut.*, Barcelona, Spain, 2005.
45. 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.
46. D. Wettergreen, C. Thorpe, and W. Whittaker. Exploring mount erebus by walking robot. *Robotics and Autonomous Systems*, 11:171–185, 1993.
47. 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.
48. 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.
49. 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.