Multi-Step Motion Planning for Free-Climbing Robots

Tim Bretl[1], Sanjay Lall[1], Jean-Claude Latombe[2], and Stephen Rock[1]

[1] Department of Aeronautics and Astronautics, Stanford University
{tbretl,lall,rock}@stanford.edu
[2] Computer Science Department, Stanford University
latombe@cs.stanford.edu

Abstract. This paper studies non-gaited, multi-step motion planning, to enable limbed robots to free-climb vertical rock. The application of a multi-step planner to a real free-climbing robot is described. This planner processes each of the many underlying one-step motion queries using an incremental, sample-based technique. However, experimental results point toward a better approach, incorporating the ability to detect when one-step motions are infeasible (i.e., to prove disconnection). Current work on a general method for doing this, based on recent advances in computational real algebra, is also presented.

1 Introduction

1.1 Broad context

The Probabilistic-Roadmap (PRM) motion planning approach [19], or one of its variants, is widely used for planning paths through high-dimensional configuration spaces subject to multiple constraints. PRM planners, which capture connectivity by a network of local paths joining sampled configurations, can often construct feasible paths very quickly. However, the approach lacks a formal stopping criterion. Typically, planners are allowed to search for a fixed length of time, after which a motion query is declared infeasible. Because PRM planners often have a fast convergence rate (i.e., the probability that they find a path when one exists goes quickly to 1), this heuristic works fine for processing a small number of queries. But imagine we want to process many queries, each in a different configuration space subject to different constraints. Then, a fundamental question arises: how much time should be spent on each query? Too much, and time is wasted on infeasible queries; too little, and feasible queries – which could be critical – may be falsely declared infeasible.

Multi-step planning for climbing robots (as well as general multi-limbed locomotion and manipulation planning) presents exactly this challenge. By applying a basic multi-step planner to a real climbing robot (Section 4), we establish the importance of incorporating a "disconnection planner" that rejects most infeasible queries. A longer time could then be allowed for remaining queries (knowing that most feasible problems are solved quickly) and a much smaller amount of time would be wasted on infeasible queries. A portion of this paper (Section 5) describes ongoing work on this topic – at this stage, we still do not have a completely satisfactory solution.

1.2 The climbing problem

Our goal is to enable a wide class of multi-limbed robots to climb vertical rock using only natural features and friction for upward progress (i.e., to *free-climb*). Such robots need not incorporate special fixtures or tools, nor need they be designed specifically for climbing, as have previous climbing robots (e.g., [2,3,11,16,17,25]). Their flexibility could benefit several application areas, including search-and-rescue, surveillance, and planetary exploration.

We consider robots with a small number of articulated limbs. We do not distinguish between limbs, and call the end-point of each one a *hand*. To climb vertical terrain, a robot must go through a continuous sequence of configurations satisfying certain constraints (e.g., equilibrium, collision, joint-torque limits). At each configuration, some of the robot's hands are in contact with the terrain – a surface with small, arbitrarily distributed features (e.g., protrusions or holes) called *holds*. During a *one-step* motion, the robot brings one hand to a new hold while using frictional contacts at other hands and internal degrees of freedom (DOF's) to maintain equilibrium. A *multi-step* motion is a sequence of one-step motions.

This paper presents a new framework for computing multi-step climbing moves. We view the multi-step motion space of the robot as a discrete graph. Corresponding to each association of hands to holds (i.e., each *stance*) is a set of feasible configurations (satisfying all constraints), that might have one or more components. Each node in the graph represents one of these components. Each edge is a one-step motion.

It is computationally practical neither to explicitly construct the feasible space at every stance, nor to exactly compute one-step motions to neighboring stances. Instead, we use an augmented PRM approach, based on our previously developed one-step planner [8,9]. Consequently, as described above, we are faced with the challenge of deciding how much time should be spent searching for each one-step motion.

2 Related Work

2.1 Multi-step planning

Multi-step motion planning for climbing robots is related to general multi-limbed locomotion and manipulation planning. Just as for climbing robots, multi-limbed locomotion is characterized by a sequence of continuous one-step movements in which feet are placed on or removed from the terrain. Likewise, manipulation paths are composed of sequences of one-step motions either of a grasped object and its manipulator or of the manipulator alone, respectively called “transfer” and “transit” motions [1].

Three main approaches to multi-step motion planning have been used:

Continuous approach. Here, the motion space is viewed as a single, high-dimensional, continuous space. The configuration of the robot moves on successive manifolds embedded in this space. Each manifold corresponds to a different set of contact constraints. This type of representation (called “stratified” [14,15] or “hybrid mechanical” [10]) is convenient for a controllability-based analysis. In particular, it leads to techniques that transform nominal paths to feasible, gaited trajectories, similar to methods used for smooth, nonholonomic systems [22,24]. However, the computational cost of these methods grows quickly with the dimensionality, geometric complexity, and number of embedded manifolds. They seem impractical for free-climbing robots.

Constraint relaxation. These methods generate an initial path considering only some of the constraints and iteratively refine this path into a feasible one by incorporating additional constraints [13]. Foot or finger placement concepts may be explicitly used in the construction of the paths. One technique proposed for manipulation planning is to first plan a trajectory for the grasped object ignoring manipulators, then generate manipulator trajectories to achieve necessary re-grasps [20]. A similar strategy for multi-limbed robots on uneven terrain is to first generate a feasible path for the center of mass (ignoring foot placement), then construct specific foot placement and limb motions that keep the center of mass stable along this path [12]. This approach can generate multi-step motions quickly for some systems. However, it is not guaranteed to find a path if one exists, and generally requires tuned heuristics to perform well. In addition, it is based on having some way of prioritizing the constraints. So, it does not handle interdependent constraints well (e.g., self-collision, equilibrium, and torque limits in climbing motions).

Foot-placement or grasp-based. These methods, like our own, carefully consider where a robot or manipulator should place its feet, hands, or fingers. Many are based on the framework for manipulation planning proposed by [1], which (as we do) views the multi-step motion space as a graph. This approach has two distinct advantages for systems such as free-climbing robots. First, by viewing the manifold associated with each set of contact constraints as a separate configuration space with its own lower-dimensional parameterization, the dimension of each one-step motion query is reduced. Second, by taking advantage of an intermediate discretization of the environment into specific foot/hand/finger placements, regions of useless contact locations are eliminated, again reducing overall problem complexity. On the other hand, the challenge of this approach is to efficiently compute the structure of the multi-step graph.

For specific systems (e.g., "spider-robots" either walking on horizontal terrain [7] or with fixed gaits in planar tunnels [29,32], and 3-limbed, planar, climbing robots subject to equilibrium constraints only [8]), it is possible to quickly compute the exact structure of this graph. For more general systems (including the climbing robots considered in this paper), it seems impractical to compute the structure of the multi-step motion space exactly. Heuristic search techniques have been used in cases where a predefined set of one-step moves is available [5,21]. If no such set has been predefined, continuous exploration is needed [26,30]. The "fuzzy PRM" approach of [26] is particularly interesting, since it seeks to address, statistically, the problem of where to allocate planning time over possible one-step motions. However, it still does not solve the fundamental problem of when to stop searching for a particular one-step motion.

2.2 Disconnection proofs

The idea of proving that two configurations are not path-connected is not new. This idea is central to the concept of "complete" or exact motion planning, in which a feasible connecting path is returned if and only if one exists [31]. But while work on exact motion planning has provided important insight and bounds on problem complexity, it has not been practical for real applications. Instead, approximate methods such as PRM [19] are used to construct feasible paths quickly, whenever possible.

Recently, more attention has been paid to developing "disconnection" planners [4], which search for proofs that no path exists. Establishing this type of proof is difficult in general, since it involves searching for disconnecting (n-1)-D manifolds in n-D space, rather than for 1-D connecting paths. The work of [4] presents one approach, for a polyhedral robot moving among polyhedral obstacles in 3-D workspace, based on bounding volumes. The proposed technique requires knowledge of critical (i.e., narrow) regions of the workspace, and is not applicable to more general constraints. Another approach is to compute both under- and over-approximations to the connectivity of simpler sub-problems, and use these approximations to prove disconnection [18]. This strategy requires a smart method of decomposition and model-reduction, and the implementation of a complete planner.

Two ideas are common to both approaches. First, both seek to relax the original feasibility problem, so that conservative proofs of infeasibility are found in a simpler space. Second, both integrate the search for disconnection proofs with a search for connecting paths (loosely, this can be described as a primal-dual method). In Section 5, we describe a systematic method of problem relaxation and a general framework for integrating "primal" and "dual" search when an over-approximation to the feasible space can be expressed as a semialgebraic set.

3 Free-Climbing Robots

3.1 Description of robot

We have focused our work on a real free-climbing robot, LEMUR IIb, developed by the Mechanical and Robotic Technologies Group at the Jet Propulsion Laboratory [9]. This robot was designed to have a number of capabilities in addition to climbing (e.g., assembly, inspection, maintenance, transport, intervention) and to be able to traverse a variety of other types of terrain (e.g., roads, talus, dirt, urban rubble).

LEMUR IIb consists of four identical limbs attached to a circular chassis, with a total mass of 7 kg (Fig. 1). Each limb contains three revolute joints, providing two in-plane (yaw) and one out-of-plane (pitch) DOF's. Each joint has an identical drive-train, capable of a maximum continuous torque of 5.0 N-m and a maximum speed of 45 deg/s. Each end-effector is a single peg wrapped in high-friction rubber. LEMUR IIb can be field-operated, with on-board batteries, processing, and sensors (including a swiveling stereo camera pair, a 6-axis force/torque sensor at each shoulder, a 3-axis accelerometer, and joint angle encoders).

The terrain climbed by LEMUR IIb in our tests is an indoor, near-vertical, planar surface. This surface is covered with small, artificial rock features exactly as are indoor "climbing gyms" for human climbers. In order to focus on the planning algorithm, the location and friction characteristics of each feature are identified manually and pre-surveyed. As a further simplification, in our current implementation we maintain LEMUR IIb's chassis parallel to and at a fixed distance from the climbing surface, and use the out-of-plane DOF in each limb only to make or break contact with features.

Finally, at this stage in our research, the robot's motion is slow enough to be assumed quasi-static.

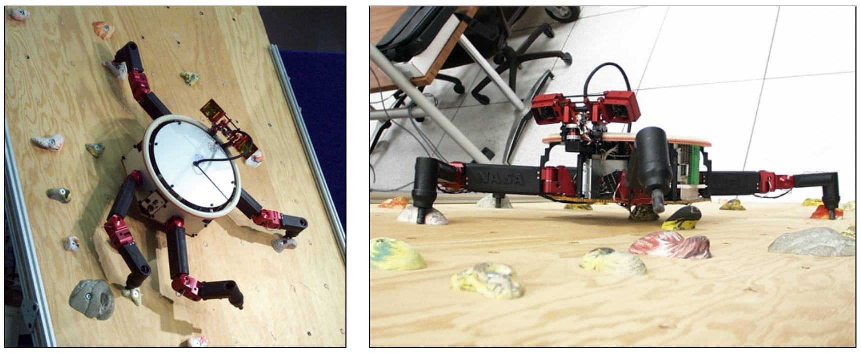

Fig. 1. LEMUR IIb, a multi-use robot capable of free-climbing (Mechanical and Robotic Technologies Group, Jet Propulsion Laboratory) [9].

3.2 Model and notation

We call the robot's circular chassis the *pelvis*. In each limb, the first joint (nearest the pelvis) is called the *shoulder*, the second joint is called the *elbow*, and the third (out-of-plane) joint is called the *wrist*. Assuming that the pelvis moves at a fixed distance parallel to the wall, any configuration of the robot is defined by 15 parameters: the position/orientation (x_p, y_p, θ_p) of the pelvis and the joint angles $(\theta_1, \theta_2, \theta_3)$ of each limb.

Any point on the terrain is a potential contact – either on the contour of a continuous rock feature, or on the planar climbing surface. We assume that a discrete number of useful contacts have been identified; these points are called *holds*. For LEMUR IIb, all holds lie on an inclined plane but can have arbitrary orientation, so each is defined by a 2-D point (x_i, y_i) and a 3-D direction $\boldsymbol{\nu}_i$. The endpoint of each limb is a *hand*. Holds at which hands are in contact are the *supporting holds*. We assume that supporting holds are point contacts, where friction is modeled by Coulomb's law.

When climbing, LEMUR IIb always maintains either three or four supporting holds. The set of supporting holds is a *stance*, denoted σ – to differentiate between 3-hold and 4-hold stances, we write $\sigma 3$ and $\sigma 4$. The linkage between the supporting holds – containing the pelvis and either three or four limbs – is called the *contact chain*. When only three supporting holds are used, the fourth limb is the *free limb*.

Because of the closed-chain constraint, the robot's continuous motion with four supporting holds occurs on a 3-D manifold $C_{\sigma 4}$ in the robot's configuration space. With three supporting holds, motion occurs on a 6-D manifold $C_{\sigma 3}$. This motion is subject to four additional constraints: quasi-static equilibrium, joint angle limits, joint torque limits, and (self-)collision. The *feasible space* at a stance σ is the subset F_σ of C_σ satisfying each of these constraints. The limbs have non-negligible mass, so their motion affects the robot's equilibrium. If two points in F_σ are connected by a continuous path in F_σ, we say they are in the same *component* of F_σ. (Henceforth, a "continuous path" will always be taken to mean a continuous motion of the robot at some fixed stance, i.e., with fixed holds.)

3.3 Climbing motions

To climb upward, the robot must switch between 3-hold and 4-hold stances. Two stances $\sigma 3$ and $\sigma 4$ are *adjacent* if $\sigma 4 = \sigma 3 \cup \{i\}$ for some hold i. The robot can only switch between adjacent stances σ and σ' (i.e., place or remove a hand) at points $q_t \in F_\sigma \cap F_{\sigma'}$. We call such points *transition points*. Given a start configuration $q_s \in F_\sigma$ at a stance σ, we say that a component of the feasible space $F_{\sigma'}$ at an adjacent stance σ' is *reachable* if there is a continuous path connecting q_s to a transition point in that component. This path is a *one-step motion*. Examples of one-step motions are shown in Figs. 2 and 3.

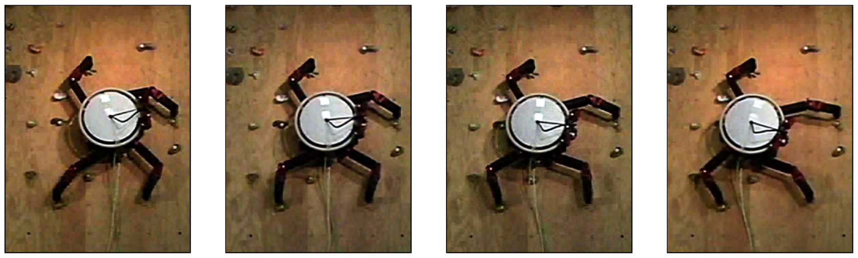

Fig. 2. A one-step motion with a 4-hold stance, to remove the bottom right hand.

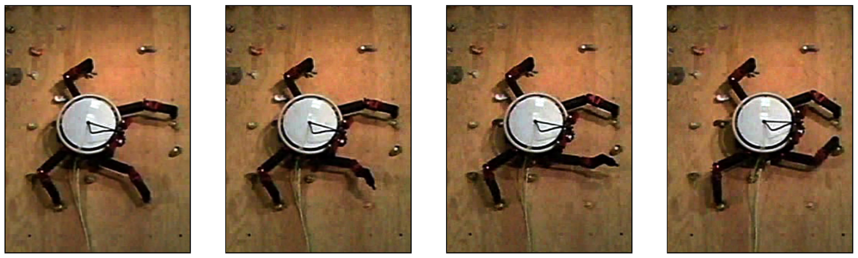

Fig. 3. A one-step motion with a 3-hold stance, to place the bottom right hand.

4 Multi-Step Planning

4.1 Basic concepts

Given a stance σ, a start configuration $q_s \in F_\sigma$, and a goal hold g, the multi-step planning problem is to construct a sequence of one-step motions that will bring the robot to a stance σ_g that contains g. (There may be a number of different valid stances σ_g.) This problem is a search through a graph. Each node in this graph is a component of F_σ at some stance σ. Each edge is a one-step motion. The sequence of one-step motions is not assumed to be periodic (i.e., a gait) – the order in which limbs are placed and removed from holds is arbitrary, and is selected as part of the graph search.

In practice, only local sensing will be available, so long multi-step plans will not be computed. Instead, short (4-10 one-step motions) plans will be generated repeatedly – with graph searches of high breadth but low depth – as the robot explores more of the terrain. However, in our current implementation the terrain has been pre-surveyed, and consequently our tests are for multi-step searches of lower breadth and higher depth.

The concept that nodes in the graph are, fundamentally, components of feasible spaces and not particular configurations is an important one. For example, one-step motions that take a configuration $q_s \in F_\sigma$ to two transition points q_{t_1}, q_{t_2} in the same component of an adjacent feasible space $F_{\sigma'}$ are equivalent. Keeping both one-step motions, and considering both points

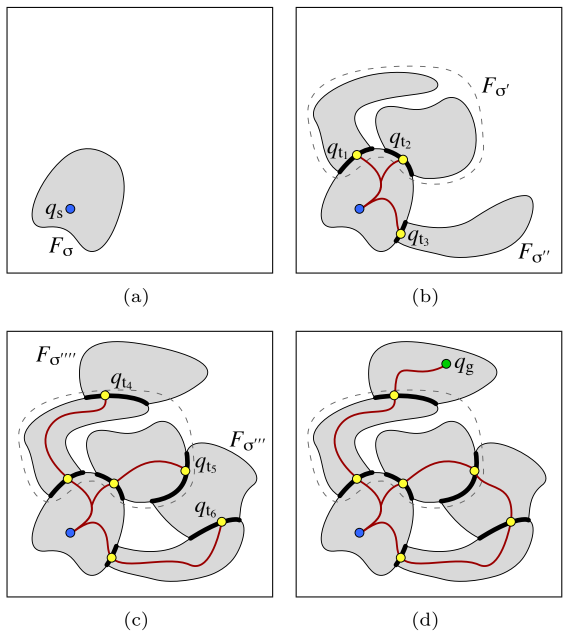

Fig. 4. Schematic of a multi-step search. Gray regions are feasible spaces at different stances (a dotted line shows the two components of $F_{\sigma'}$). Bold arcs are intersections between feasible spaces at adjacent stances. Thin lines are one-step motions.

q_{t_1} and q_{t_2} as distinct, adds redundant nodes and edges to the graph. This increases the overall planning time of a multi-step search. So although it might be convenient to index nodes by individual configurations and edges by point-to-point paths, these always represent entire components and the transitions between them.

Figure 4 shows a schematic of how a multi-step search might proceed. At each stage, transition points are found to adjacent stances at which components of the feasible space $F_{\sigma'}$ intersect the base feasible space F_σ. Note that the feasible space at stance σ' has two components, each of which admits one-step motions to different adjacent stances. Eventually, a path is constructed to a configuration q_g in the interior of the feasible space $F_{\sigma''''}$ at a stance containing the goal hold.

4.2 Consequences of an incremental sample-based approach

In general, the graph is much larger – and many more one-step queries must be made – than in the previous example. It is computationally impractical to determine exactly which adjacent stances are reachable from a start configuration $q_s \in F_\sigma$. Therefore, we use an approximate method: for each adjacent stance σ', first we sample transition points $q_t \in F_\sigma \cap F_{\sigma'}$, then we try to construct a continuous path from q_s to each q_t, using an augmented PRM planner [9]. Similar techniques are used in other multi-step planning algorithms (e.g., [21,26,30]).

However, this approach has two consequences. First, since the structure of each feasible space F_σ is never exactly known, redundant nodes and edges may be introduced. As noted in Section 4.1, this redundancy increases overall planning time. Second, since transition points are sampled and feasible paths are found only with some probability, nodes and edges may be missed. This reduces the overall probability of success and tends to increase the resulting path length (if a path is found).

4.3 Baseline results

Long multi-step paths. We applied our one-step planner to generate climbing motions for LEMUR IIb, which were subsequently executed by the real robot. In these initial tests, only one-step moves were computed by the planner. To get multi-step paths, the user specified which hold to grab or release at each step. Snapshots from one climb, taking the robot from bottom to top of the climbing surface, are shown in Fig. 5. The one-step planner generated, on-line, each of 88 one-step motion trajectories forming the multi-step path.

The distribution of planning times for this example is shown in Fig. 6. Using our sampling scheme, most one-step moves were planned very quickly (less than 0.5 s). However, several difficult moves took more time (more than 5.0 s), even as much as 17.3 s. This illustrates the important issue raised in Section 1.1. Many one-step motion queries are made in a multi-step search; how should we choose the amount of time T_{max} after which the one-step

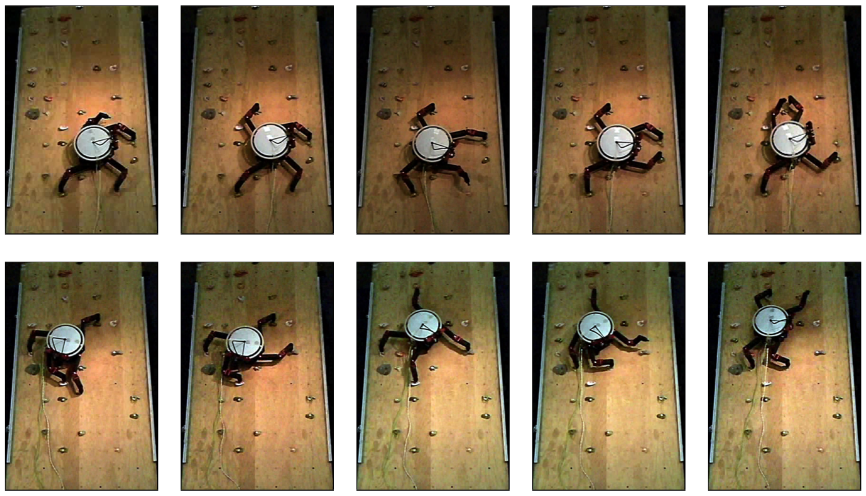
Fig. 5. Snapshots of a multi-step climb.

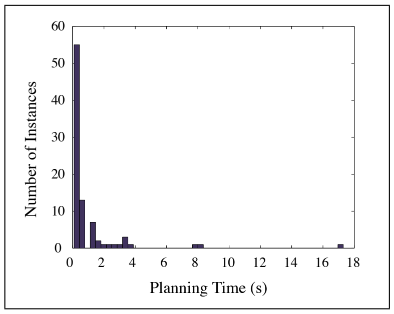

Fig. 6. The distribution of one-step planning times along a multi-step path consisting of 88 one-step motions. Minimum planning time was 0.09 s, maximum was 17.3 s. Mean planning time was 1.02 s, but over 75% of the one-step motions were computed quicker than average.

planner returns failure? In this example, suppose $T_{max} = 2.0\ s$ (twice the mean value) – infeasible one-step motions are rejected quickly, but several difficult moves (15% of the multi-step path) likely are not found. Alternatively, suppose $T_{max} = 20.0\ s$ (greater than the maximum value) – now, all feasible one-step motions likely are found, but every infeasible query takes ten times longer, drastically increasing total search time.

In this example, all one-step motions were computed along a single, user-defined, multi-step path. In a general search, many different multi-step paths may exist between start and goal stances. However, a "critical passage" in the multi-step graph is not necessarily a single critical edge (i.e., a single, difficult, one-step move). If the graph can be divided into two components – one containing the start node, the other containing the goal – such that each edge between these two components represents a one-step motion that is difficult to compute, then this group of edges represents a critical passage.

Performance of a basic multi-step planning algorithm. We have also implemented a fully automatic, multi-step planner for LEMUR IIb, incorporating the same one-step planning algorithm. This planner does not yet exploit search heuristics or other augmentations (e.g., our disconnection proof of Section 5). We have tested our planner in simulation, to understand the effects of varying the amount of time T_{max} it is allowed to search for each one-step motion. Figure 7 shows the results of one experiment. In this example, a goal hold was selected at a distance of eight one-step motions from the robot's initial stance. Results of multi-step planning were averaged over 100 runs at each T_{max}. (In our experiments, it was convenient to bound T_{max} implicitly, e.g., by bounding numbers of iterations – therefore, results are given with respect to mean planning time rather than T_{max} in Fig. 7.)

Both total path length and the ratio of edges to nodes in the graph asymptotically reach minimum and maximum values, respectively, as T_{max} is increased. This trend arises because the structure of the graph is approximated increasingly well (i.e., components of each F_σ are better explored, and feasible one-step motions are found with higher probability). It also indicates why total planning time approaches linear growth with T_{max} – the number of nodes and edges explored in the multi-step search becomes constant.

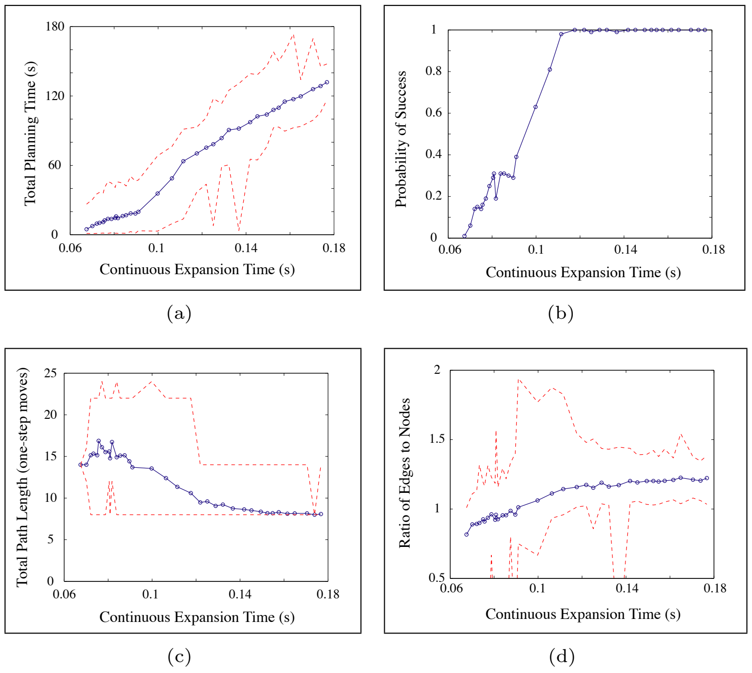

Fig. 7. Performance of a basic multi-step search using PRM, as the average time allowed for continuous expansion is varied. Mean values (over many runs) are shown; bounds are dotted lines. (**a**) Total planning time. (**b**) Probability of success. (**c**) Total number of one-step motions in the final path. (**d**) Ratio of edges to nodes.

4.4 Proposed modifications

There are a number of important modifications we intend to make to our planner in order to improve its performance (e.g., the use of graph search heuristics, and a "lazy" approach to checking the feasibility of edges). However, as shown by our baseline results, a fundamental problem that still must be addressed is to determine when one-step motions are infeasible. We propose a possible, general method for doing so in the following section.

We intend to apply our "disconnection planner" as follows: given each one-step motion query, our sample-based planner will be allowed to run for a short length of time, say 1.0 s. As shown in Fig. 6, many feasible one-step motions will be found immediately. When the one-step planner is unable to find a solution, we will attempt to prove that the motion is infeasible. This will eliminate a large number of infeasible queries quickly. Finally, if no disconnection proof is found, we allow our one-step planner a further amount of time T_{max} sufficient to find feasible one-step motions with high probability. In this way, the fraction of queries processed for time T_{max} will be significantly reduced.

5 Proving One-Step Disconnection

5.1 General approach

Our goal is to show that no feasible, continuous path exists between two configurations $q_s, q_g \in F_\sigma$. One way of establishing such a "disconnection proof" is to construct a manifold embedded in the infeasible space $C_\sigma - F_\sigma$ that separates q_s and q_g. Such a manifold can be defined as the zero-level-set of a continuous, scalar-valued function $g : C_\sigma \to \mathbb{R}$ such that $g(q_s) > 0$ and $g(q_g) < 0$, where we require that $\{q \in F_\sigma | \, g(q) = 0\}$ is empty. The value of g must be zero at some configuration q_i along any continuous path between q_s and q_g, but by definition any such q_i is infeasible. Therefore, the function g is a proof that no feasible, continuous path exists between q_s and q_g.

This approach need not involve the construction of a complete roadmap for F_σ. Indeed, it turns a question about the existence of paths into a question about the feasibility of sets. In certain cases, the latter computation can be done quickly. In this section, we propose a method that applies to the case where F_σ is expressed as a semialgebraic set. We do not suggest the use of exact arithmetic – instead, our method is based on a hierarchy of *Positivstellensatz* refutations, each of which can be computed as a semidefinite program (a result from the fields of convex optimization and real algebra [27,28]).

5.2 Theoretical foundation

Problem statement. Assume the feasible space F_σ can be represented as a *semialgebraic set.* (An over-approximation $\hat{F}_\sigma$ of F_σ is sufficient. If $F_\sigma \subset \hat{F}_\sigma$, then empty $\{q \in \hat{F}_\sigma | \, g(q) = 0\}$ implies empty $\{q \in F_\sigma | \, g(q) = 0\}$.) So, it can be represented as the following subset of an n-dimensional Euclidean space $\mathbb{R}^n$, where all $f_i, h_j \in \mathbb{R}[x_1, \dots, x_n]$ are polynomials:

$$F_\sigma = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} f_i(x) \geq 0 & i = 1, \dots, m_f \\ h_j(x) = 0 & j = 1, \dots, m_h \end{array} \right\} \tag{1}$$

The disconnection problem is to find a polynomial function $g \in \mathbb{R}[x_1, \dots, x_n]$ such that $g(q_s) > 0$, $g(q_g) < 0$, and the following set $P_{\text{cut}}(g)$ is empty:

$$P_{\text{cut}}(g) = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} f_i(x) \geq 0 & i = 1, \dots, m_f \\ h_j(x) = 0 & j = 1, \dots, m_h \\ g(x) = 0 & \end{array} \right\} \tag{2}$$

This problem involves two basic challenges. First, given a function g, we must be able to check that $P_{\text{cut}}(g)$ is empty. Second, we must be able to search for such a function g.

Proving that $P_{\text{cut}}(g)$ is empty. A general strategy for showing the infeasibility of $P_{\text{cut}}(g)$ is to assume feasibility and derive a contradiction. When F_σ is semialgebraic, it is natural to begin with the hypothesis that there exists $x_0 \in P_{\text{cut}}(g)$ and try to infer a false conclusion such as $-1 \geq 0$. This is *automated proof by contradiction*, using algebra to perform inference.

For example, consider a feasible space F_σ defined by the single inequality $f_1(x) = x^2 - 1$, so $F_\sigma = (-\infty, -1] \cup [1, \infty) \subset \mathbb{R}$. To separate points $q_s = 2$, $q_g = -2$, we could use the function $g(x) = x$, which defines a manifold that is the single point $\{0\}$. We have $g(q_s) = 2 > 0$ and $g(q_g) = -2 < 0$, so all that remains is to show that $P_{\text{cut}}(g)$ is empty. But by definition, $f_1 \geq 0$ and $g = 0$ on $P_{\text{cut}}(g)$, so by inference $-1 = f_1 + (-x) \cdot g = f_1 + (-x) \cdot 0 = f_1 \geq 0$ on $P_{\text{cut}}(g)$, so if $P_{\text{cut}}(g)$ is nonempty we have a contradiction.

In fact, there is a systematic way to search for such a contradiction. The set of all polynomials that can be inferred from $f_i \geq 0 \ \forall i$ to be nonnegative on $P_{\text{cut}}(g))$ is called the *cone* generated by $f_1, \ldots, f_{m_f}$, and is denoted $\mathbf{cone}\{f_1, \ldots, f_{m_f}\}$. Equivalently, the set of all polynomials that can be inferred from $h_j = 0 \ \forall j$ and $g = 0$ to be zero on $P_{\text{cut}}(g)$ is called the *ideal* generated by $h_1, \ldots, h_{m_h}, g$, and is denoted $\mathbf{ideal}\{h_1, \ldots, h_{m_h}, g\}$. The *Positivstellensatz* [33] states that $P_{\text{cut}}(g)$ is empty if and only if

$$-1 \in \mathbf{cone}\{f_1, \ldots, f_{m_f}\} + \mathbf{ideal}\{h_1, \ldots, h_{m_h}, g\} \tag{3}$$

As described in [6], the cone and the ideal can be parameterized so (3) is exactly equivalent to the existence of polynomials r and t_i and sum-of-squares polynomials $s_0, s_i, s_{ij}, \ldots,$ (all of arbitrary degree) such that the following algebraic equation holds (note that the series does terminate):

$$-1 = \left(s_0 + \sum_{i=1}^{m_f} s_i f_i + \sum_{i=1}^{m_f} \sum_{j=1}^{m_f} s_{ij} f_i f_j + \ldots \right) + \left(\sum_{i=1}^{m_h} t_i h_i + rg \right) \tag{4}$$

Recently, it has been shown that for fixed degree r, t_i, and $s_0, s_i, s_{ij}, \ldots,$ the feasibility of (4) can be checked by solving an SDP [27]. The basic idea is to search over the coefficients of these polynomials, restricting the coefficients of $s_0, s_i, s_{ij}, \ldots,$ to lie within the positive semidefinite cone. This approach allows the use of floating-point computation, rather than exact algebraic manipulation. A hierarchy of checks is also possible, with polynomial multipliers of higher and higher degree, in which each SDP is a better approximation of the "only if" condition of (4). However, the SDP's become significantly larger and more costly to compute. In practice, a degree is usually fixed (in our experiments, low degree certificates have been sufficient).

Searching for a function g. In some cases good candidate functions g are already available. In fact, this is true for the climbing robot problem, where F_σ is often disconnected across manifolds corresponding to $\theta_2 = 0$ in one limb (i.e., a straight-limb configuration). If candidates are not available, directly incorporating the search for g into the SDP computation (4) is problematic.

However, there is a duality theory associated with the *Positivstellensatz* and its corresponding SDP's [23]. This means that if both primal (connection) and dual (disconnection) searches are conducted in parallel, information from each can be used to help the other (also as suggested in [4]). For example, if $P_{\text{cut}}(g)$ is not empty, there must exist some feasible configuration $q_f \in P_{\text{cut}}(g)$. This can be used to direct the primal search – if the primal algorithm is sample-based, samples can be drawn from a distribution around nonempty $P_{\text{cut}}(g)$. Likewise, candidate functions g can be postulated by fitting hypersurfaces to infeasible sampled configurations $q_i \notin F_\sigma$.

5.3 Preliminary application

To test how this algebraic method might be applied, we used it to prove that the feasible space at the 4-hold stance σ in the last frame of Fig. 3 has at least two components. In this example, F_σ can be separated along a manifold corresponding to $\theta_2 = 0$ for the bottom-left limb. So we can prove that no continuous path exists between any q_s at which $\theta_2 > 0$ and any q_g at which $\theta_2 < 0$ in this limb.

The feasible space F_σ is over-approximated by considering only the closed-chain and joint-angle constraints. There are a variety of possible parameterizations to express F_σ as a semialgebraic set; the choice is important, since it may result in different numbers of variables, numbers of constraints, or total polynomial degrees of the constraints. In this case it is convenient to use variables x_{mi}, y_{mi} and x_{si}, y_{si} defining the locations of the elbow and shoulder joints, respectively, of each limb i.

Equality constraints h_j are introduced to enforce the spatial relationship between joints and the closed-chain constraint between the supporting holds. Inequality constraints f_i are used to enforce the joint-angle limits, which can be represented as minimum distances between joints. The function g can be expressed implicitly by constraining the shoulder of the bottom-left limb to be a fixed distance (corresponding to $\theta_2 = 0$) from its supporting hold. After eliminating redundancy, the set $P_{\text{cut}}(g)$ can be defined by 8 equality constraints and 11 inequality constraints, in 10 variables.

A certificate of the form (4) with fixed total degree 4 (i.e., a proof that no path exists in F_σ that crosses $\theta_2 = 0$ in the bottom-left limb) was found in 90.0 s on a 1GHz PowerPC G4. Our current implementation is still not fast enough to be practical (e.g., compare with the planning times we presented in Section 4.3). However, we have generated much faster (less than 1.0 s) disconnection proofs for a variety of simpler systems, including a 3-limbed planar robot. We hope to reduce the running time for this application by better exploiting the sparsity of the SDP's (e.g., [28]). Many other such improvements may be possible – this is ongoing work.

6 Future Work

This paper presented a framework for non-gaited, multi-step motion planning for free-climbing robots. Based on experimental results, it showed the need to detect when one-step motions are infeasible (i.e., to prove disconnection). It also presented current work on the development of a general method for doing so, based on advances in computational real algebra.

Other challenging problems remain to be addressed – sensing, grasping, and control are prominent among them. For example, here a pre-surveyed model of the terrain was assumed. With only local sensing, the robot will have to explore the environment incrementally, and plan based on incomplete or uncertain information. Also, there are many unanswered questions raised by our treatment of the disconnection problem. Work on exact motion planning has provided important insight, but to date, few practical algorithms. The use of floating-point computational tools, rather than exact arithmetic, could make it possible to implement some of these algorithms more efficiently.

Acknowledgments. The authors would like thank E. Baumgartner, B. Kennedy, and H. Aghazarian of the Mechanical and Robotic Technologies Group at JPL for their help and support.

References

1. R. Alami, J.-P. Laumond, and T. Simeon. 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.
2. M. Almonacid, R. Saltarén, R. Aracil, and O. Reinoso. Motion planning of a climbing parallel robot. *IEEE Tr. Rob. Aut.*, 19(3):485–489, 2003.
3. C. Balaguer, A. Giménez, J. Pastor, V. Padrón, and M. Abderrahim. A climbing autonomous robot for inspection applications in 3d complex environments. *Robotica*, 18:287–297, 2000.
4. J. Basch, L. Guibas, D. Hsu, and A. T. Nguyen. Disconnection proofs for motion planning. In *IEEE Int. Conf. Rob. Aut.*, pages 1765–1772, Seoul, Korea, 2001.
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.*, volume 4, pages 4009–4014, 2000.
6. J. Bochnak, M. Coste, and M.-F. Roy. *Real Algebraic Geometry.* Springer, 1998.
7. J.-D. Boissonnat, O. Devillers, and S. Lazard. Motion planning of legged robots. *SIAM J. Computing*, 30(1):218–246, 2000.
8. T. Bretl, J.-C. Latombe, and S. Rock. Toward autonomous free-climbing robots. In *Int. Symp. Rob. Res.*, Siena, Italy, 2003.
9. T. Bretl, S. Rock, J.-C. Latombe, B. Kennedy, and H. Aghazarian. Free-climbing with a multi-use robot. In *Int. Symp. Exp. Rob.*, Singapore, 2004.
10. F. Bullo and M. Žefran. Modeling and controllability for a class of hybrid mechanical systems. *IEEE Tr. Rob. Aut.*, 18(4):563–573, 2002.
11. H. Dulimarta and R. L. Tummala. Design and control of miniature climbing robots with nonholonomic constraints. In *WCICA*, Shanghai, P.R.China, 2002.
12. C. Eldershaw and M. Yim. Motion planning of legged vehicles in an unstructured environment. In *IEEE Int. Conf. Rob. Aut.*, Seoul, South Korea, 2001.

13. P. Ferbach and J. Barraquand. A method of progressive constraints for manipulation planning. *IEEE Tr. Rob. Aut.*, 13(4):473–485, 1997.
14. B. Goodwine and J. Burdick. Controllability of kinematic control systems on stratified configuration spaces. *IEEE Tr. Aut. Cont.*, 46(3):358–368, 2001.
15. B. Goodwine and J. Burdick. Motion planning for kinematic stratified systems with application to quasi-static legged locomotion and finger gaiting. *IEEE Tr. Aut. Cont.*, 18(2):209–222, 2002.
16. J. C. Grieco, M. Prieto, M. Armada, and P. G. de Santos. A six-legged climbing robot for high payloads. In *IEEE Int. Conf. Cont. App.*, Trieste, Italy, 1998.
17. S. Hirose, A. Nagabuko, and R. Toyama. Machine that can walk and climb on floors, walls, and ceilings. In *ICAR*, pages 753–758, Pisa, Italy, 1991.
18. S. Hirsch and D. Halperin. Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane. In *WAFR*, pages 225–241, Nice, France, 2002.
19. L. E. Kavraki, P. Svetska, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Tr. Rob. Aut.*, 12(4):566–580, 1996.
20. Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *IEEE Int. Conf. Rob. Aut.*, volume 2, pages 945–952, San Diego, CA, 1994.
21. 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.
22. G. Lafferriere and H. Sussmann. A differential geometric approach to motion planning. In Z. Li and J. Canny, editors, *Nonholonomic Motion Planning*, pages 235–270. Kluwer, 1993.
23. J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Opt.*, 11(3):796–817, 2001.
24. J. Laumond, P. Jacobs, M. Taix, and R. Murray. A motion planner for nonholonomic mobile robots. *IEEE Tr. Rob. Aut.*, 10(5):577–593, 1994.
25. W. Neubauer. A spider-like robot that climbs vertically in ducts or pipes. In *IROS*, pages 1178–1185, Munich, Germany, 1994.
26. C. L. Nielsen and L. E. Kavraki. A two level fuzzy prm for manipulation planning. In *IEEE/RSJ IROS*, pages 1716–1721, Takamatsu, Japan, 2000.
27. P. A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, Pasadena, CA, 2000.
28. P. A. Parrilo and S. Lall. Semidefinite programming relaxations and algebraic optimization in control. *European Journal of Control*, 9(2-3):307–321, 2003.
29. E. Rimon, S. Shoval, and A. Shapiro. Design of a quadruped robot for motion with quasistatic force constraints. *Autonomous Robots*, 10:279–296, 2001.
30. A. Sahbani, J. Cortés, and T. Siméon. A probabilistic algorithm for manipulation planning under continuous grasps and placements. In *IEEE/RSJ IROS*, pages 1560–1565, Lausanne, Switzerland, 2002.
31. J. Schwartz and M. Sharir. On the 'piano movers' problem II: General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4(1):298–351, 1983.
32. 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.
33. G. Stengle. A nullstellensatz and a positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207:87–97, 1974.