Motion Planning of Multi-Limbed Robots Subject to Equilibrium Constraints: The Free-Climbing Robot Problem

Timothy Bretl

January 3, 2006

Abstract

This paper addresses the problem of planning the motion of a multi-limbed robot in order to "free-climb" vertical rock surfaces. Free-climbing only relies on frictional contact with the surfaces rather than on special fixtures or tools like pitons. It requires strength, but more importantly it requires deliberate reasoning: not only must the robot decide how to adjust its posture to reach the next feature without falling, it must plan an entire sequence of steps, where each one might have future consequences. In this paper, this process of reasoning is broken into manageable pieces by decomposing a free-climbing robot's configuration space into manifolds associated with each state of contact between the robot and its environment. A multi-step planning framework is presented that decides which manifolds to explore by generating a candidate sequence of hand and foot placements first. A one-step planning algorithm is then described that explores individual manifolds quickly. This algorithm extends the probabilistic roadmap approach to better handle the interaction between static equilibrium and the topology of closed kinematic chains. It is assumed throughout this paper that a set of potential contact points has been presurveyed. Validation with real hardware was done with a four-limbed robot called LEMUR (developed by the Mechanical and Robotic Technologies Group at NASA-JPL). Using the planner presented in this paper, LEMUR free-climbed an indoor, near-vertical surface covered with artificial rock features.

1 Introduction

Imagine a human climber, nearing the top of a desolate mountain peak. He hangs by his fingertips from a notch in the sheer rock face, one foot pasted to an incut edge, the other dangling over two thousand feet of air. He is "free-climbing," and carries no hooks or pitons to aid his ascent. A rope, attached to the harness at his waist, runs down through several pieces of "protection" to his partner below, but this offers little reassurance. He pauses, examining the cliff above him to choose a sequence of hand- and foot-holds, picturing his motion through crux sections: shift weight to one side of a vertical arete; press down with both palms to surmount a bulge. Carefully, step by step, he inches his way upward, always looking ahead to plan his route.

Figure 1: Free-climbing requires coordination and balance. Only frictional contact at a carefully chosen set of holds allows the climber to avoid falling.

What if a multi-limbed robot could autonomously free-climb vertical rock walls, just like the human climber? Locating stranded mountaineers, scrambling over broken urban terrain to help in search-and-rescue operations, exploring cliff faces of scientific interest on the Moon and Mars – these activities are useful and exciting, but they also present a new level of challenge for robotics. For example, it is absolutely critical for a human free-climber to think through his moves before performing them. The same is true for a robotic free-climber. Not only must it select a sequence of holds on which to climb (where a bad choice might lead to a dead-end), it must decide how to adjust its posture to reach each hold without falling (where a bad choice reaching one might make it impossible to reach the next). This process of reasoning is the topic of our paper.

1.1 Why is it hard to plan free-climbing motions?

The only thing that keeps a free-climber from falling is frictional contact with a carefully chosen set of *holds* – natural features such as holes or protrusions. The climber must apply contact forces at these holds that exactly compensate for gravity without causing hands or feet to slip (Fig. 1). In order to move upward, he must adjust his internal posture to maintain contact and equilibrium while reaching for a new hold. In addition to strength, this process requires precise coordination and balance. For a free-climbing robot, this means reasoning about many interdependent constraints (such as equilibrium, closed-chain kinematics, collision-avoidance, and torque limits), all of which affect it differently at each set of holds on which it stands.

At an abstract level, we are simply asking the robot to plan its own motions, a central and well-studied problem of robotics [15, 30]. For many years, the classic challenge in motion planning has been to generate the collision-free path of a rigid object or multi-link arm through a physical workspace containing a number of obstacles. However, the structure of a free-climbing robot's configuration space is quite different. To maintain contact with the terrain, the robot must coordinate the motion of each limb to satisfy a set of constraints (for instance, closed-loop kinematic constraints that ensure limb endpoints in contact with the terrain do not move). Consequently, its configuration is restricted to lie in a lower-dimensional surface or *manifold* in configuration space. A different such manifold is associated with each set of holds. So, grabbing or releasing a hold (taking a *step*) corresponds to switching between manifolds. Other constraints are also specific to the set of holds (equivalently, to the corresponding manifold). For example, the robot might be able to balance at a particular configuration while standing on a particular set of holds, but not if it releases one of them. In fact, each manifold can be viewed as a separate configuration space, with a different parameterization and subject to different constraints.

So, motion planning for free-climbing robots involves two levels of difficulty. Planning a single step (*one-step planning*) is hard – there are several constraints to consider, they may define a geometrically complex feasible region in the corresponding manifold, and this region might contain narrow passages. But planning a sequence of steps (*multi-step planning*) is even harder – there might be many potential sequences of steps, most of which lead to dead-ends where progress is no longer possible.

1.2 What problems are related to free-climbing?

Multi-step planning is useful for many applications other than free-climbing. For example, whenever a humanoid robot walks through a room, it (literally) computes a sequence of steps to take. Of course, the humanoid probably uses a fixed gait to make planning easier, whereas free-climbing motions tend to be unique and non-gaited. However, there is no clear distinction between free-climbing and walking – according to books on human mountaineering, these two activities lie along a continuum of difficulty [33].

Free-climbing also resembles dexterous manipulation. While a multi-fingered hand grasps an object, a multi-limbed robot "grasps" its environment. To rotate an object, the hand sequentially grabs and releases finger contacts, just like the free-climber grabs and releases holds. These applications differ – for instance, the free-climbing robot must remain in equilibrium as it moves, but when fingers are re-positioned on an object, the object is what needs to remain in equilibrium. Nevertheless, many challenges are common [5, 38] (and ultimately, the "hands" of free-climbing robots may, themselves, be dexterous manipulators).

Manipulation planning is another similar application. It is distinguished from dexterous manipulation by the fact that it generally involves the rearrangement of many objects using a simple robotic manipulator. A "manipulation path" is a sequence of motions of the manipulator grasping an object (one state of contact) and of the manipulator alone (a different state of contact), just like a free-climb is a sequence of steps [2]. The configuration space considered in manipulation planning also consists of a collection of manifolds, although these manifolds are usually "flat" in the sense that when one object is being manipulated, the configuration of all other objects remains fixed.

1.3 How are free-climbing robots different from other climbing robots?

Various climbing robots have been developed previously:

- **Adhesive robots** "stick" to featureless, flat, or smoothly curved surfaces. Most use suction cups [14, 17, 35] or magnets [20]. Consequently, most are limited to environments consisting of glass or metal.
- **Robots for engineered environments** have end-effectors that match engineered features of the environment like pegs [4], fences or porous materials [46], handrails or bars [1], and poles [3].
- **Robots for pipes and ducts** rely on frictional contacts with surfaces (as do free-climbers), but take advantage of geometric regularity to use precomputed, gaited motion [36, 40].

In addition, new technologies have recently emerged that synthesize the adhesion mechanism in gecko toes and may enable robots to climb a wider class of terrain, including natural rock [44]. Each of these robots was designed to make it easy to stay attached to a vertical surface and to minimize the need for motion planning.

In this paper we take a different approach. *Rather than design a "climbing robot," we design a motion planner to enable more general multi-limbed robots to free-climb.* The robot used in our experiments, the Legged Excursion Mechanical Utility Rover (LEMUR), was designed by collaborators in the Mechanical and Robotic Technologies Group at NASA-JPL to explore cliff faces of scientific interest on the Moon, Mars, and asteroids [12]. It does not carry special fixtures or tools to grasp a rock surface; instead, each end-effector is a rigid "finger" wrapped in high-friction rubber (see Fig. 18), so LEMUR is subject to the same constraints as a human free-climber. But our planner is not specific to LEMUR, nor even to free-climbing robots. For example, it has been extended to a humanoid robot navigating severely rough (broken, sloped, or irregular) terrain [23], and recently even to the six-legged lunar robot ATHLETE (also part of a joint project with NASA-JPL).

1.4 Organization of this paper

This paper consists of the following sections:

- **Problem definition (Section 2).** This section introduces the free-climbing robot problem. We consider three example systems that are studied throughout the rest of this paper. We show that a free-climbing robot's configuration space consists of a collection of manifolds associated with each state of contact between the robot and its environment.
- **Multi-step planning framework (Section 3).** This section describes our overall strategy for planning free-climbing motions. We represent the configuration space at two levels of resolution: coarse graphs containing points of intersection between manifolds, and fine graphs containing paths in manifolds between points of intersection.

We search the former (generating a candidate sequence of hand and foot placements) to guide our search through the latter (generating continuous motions to reach them).

- **One-step planning algorithms (Section 4).** This section develops two specific algorithms embedded in our framework. We use these algorithms to explore individual manifolds, quickly finding points of intersection and paths between them. They are based on the probabilistic roadmap (PRM) approach to motion planning, but they also extend this approach to better handle the interaction between the constraint of static equilibrium and the topology of closed kinematic chains.

- **Experimental validation (Section 5).** This section describes an integrated implementation on the LEMUR robot. Using our planner (but with limited control and sensing), LEMUR free-climbed an indoor, near-vertical rock surface covered with artificial rock features. These experiments, in turn, impacted the design of our planner.

Altogether, our main contribution is an integrated planner for computing non-gaited, multi-step, free-climbing motions, bringing together several previous results [8,10–12]. A key result of our work is to show that it is possible to make a general-purpose, multi-limbed robot free-climb – despite limited control and sensing – provided that it has a suitable motion planner.

2 The free-climbing robot problem

The motion of a free-climbing robot is governed largely by two constraints: *contact* (keep hands at a carefully chosen set of holds) and *equilibrium* (apply forces at these holds that exactly compensate for gravity without causing slip). In this section we consider how these constraints shape the robot's configuration space, an understanding that is critical to the design of our planner. To do so, we introduce three example systems of increasing complexity, which will be studied throughout the rest of this paper.

2.1 Double-link robot

First, consider a simple robot composed of two rigid links connected by a revolute joint (Fig. 2). Unlike later examples, this robot does not rely on frictional contact, but rather "climbs" by grabbing and releasing pegs. However, its configuration space has the same underlying structure, and is easier to visualize.

The robot's *workspace* is a plane $\mathbb{R}^2$ that contains several pegs and scattered obstacles. We call each end of the robot a *hand*, and each peg a *hold*. The robot has no fixed base, and can move only by rotating with hands on holds, possibly adjusting its central angle, while avoiding collision with obstacles. Fig. 3 shows an example motion through a workspace with four holds and three circular obstacles (assume that this motion is both quasi-static and reversible).

A *configuration* of this robot is $q = (x, y, \theta_1, \theta_2)$, where $(x, y) \in \mathbb{R}^2$ is the position of one end of the robot, $\theta_1 \in S^1$ is the angle of the first link (the one containing the end at (x, y)) with respect to a fixed reference frame, and $\theta_2 \in S^1$ is the angle of the second link relative to the first. Its *configuration space* C is $\mathbb{R}^2 \times S^1 \times S^1$. Normally we would say that a

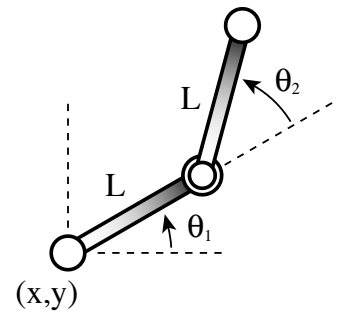


Figure 2: A double-link robot translating and rotating in a plane.

configuration is *free* if it represents a position of the robot that does not collide with any obstacles. However, as the double-link robot also has to maintain contact with holds, we say that a configuration is *feasible* if it is free and places at least one hand at a hold. The subset of all feasible points in configuration space is the *feasible space* F.

With one hand at a hold, the robot's configuration can be specified by only two parameters, the angles θ_1, θ_2. Hence, the 4-D configuration of this robot is restricted to move in 2-D subsets of configuration space. There is a different subset, a torus, for each set of holds. We will call each set of holds a *stance*, and the corresponding subset of C a *stance manifold*.[1] For example, Fig. 4 shows the stance manifold at Hold 1, where the motion of Fig. 3(b) takes place. It is depicted as a separate 2-D Cartesian feasible space F_1, parameterized by the angles θ_1, θ_2 (both taken modulo 2π, so movement past $+\pi$ wraps continuously to $-\pi$ in each direction). For a stance at a single Hold i, we take the convention that θ_1 is always the angle of the link contacting i. For a stance at two holds, two such parameterizations are possible.

All four stance manifolds (one at each hold) are shown in Fig. 5. Three have a single path-connected component (F_1, F_3, F_4) and one has four components $(F_2^1, \ldots, F_2^4)$. Some of these feasible spaces intersect – we call each intersection point a *transition*. For example, there are two configurations, q_1 and q_2, that place hands at both Holds 1 and 2, differentiated by which way the joint between the links is bending (that is, by whether $\theta_2 > 0$ or $\theta_2 < 0$). Both configurations are feasible at both stances, so they are transitions between F_1 and F_2. When the robot moves to one of these transitions, it can take a *step*, releasing Hold 1 and grabbing Hold 2 (or vice versa). In this case, the choice of transition is crucial. Although q_1 and q_2 are in the same (and only) component of F_1, they are in two distinct components of F_2, and so admit different subsequent motion.

We can trace a sequence of feasible spaces and transitions

$$F_1 \xrightarrow{q_1} F_2^1 \xrightarrow{q_3} F_3 \xrightarrow{q_4} F_2^3 \xrightarrow{q_5} F_4$$

from the robot's initial position at Hold 1 to its final position at Hold 4. For example, Fig. 6 shows the sequence of feasible paths corresponding to Fig. 3 (although many other paths might accomplish the same sequence of steps).

[1] In fact, a stance is an association of hands to holds. But because of symmetry in both the double-link robot and the three-limbed robot considered in the following section, the particular association does not matter. So for simplicity, we refer to a stance by its set of holds only.

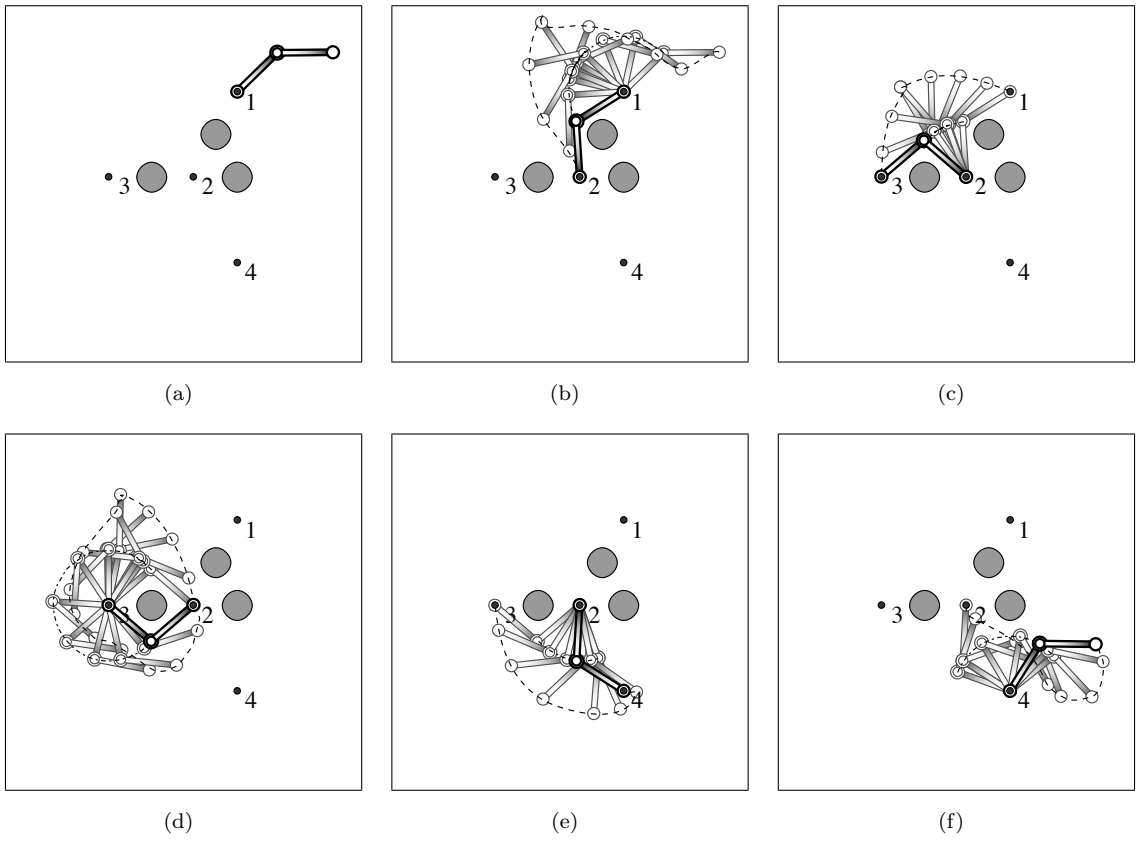


Figure 3: The robot's workspace contains four holds (numbered) and three circular obstacles. It takes a sequence of steps from Hold 1 to Hold 4.

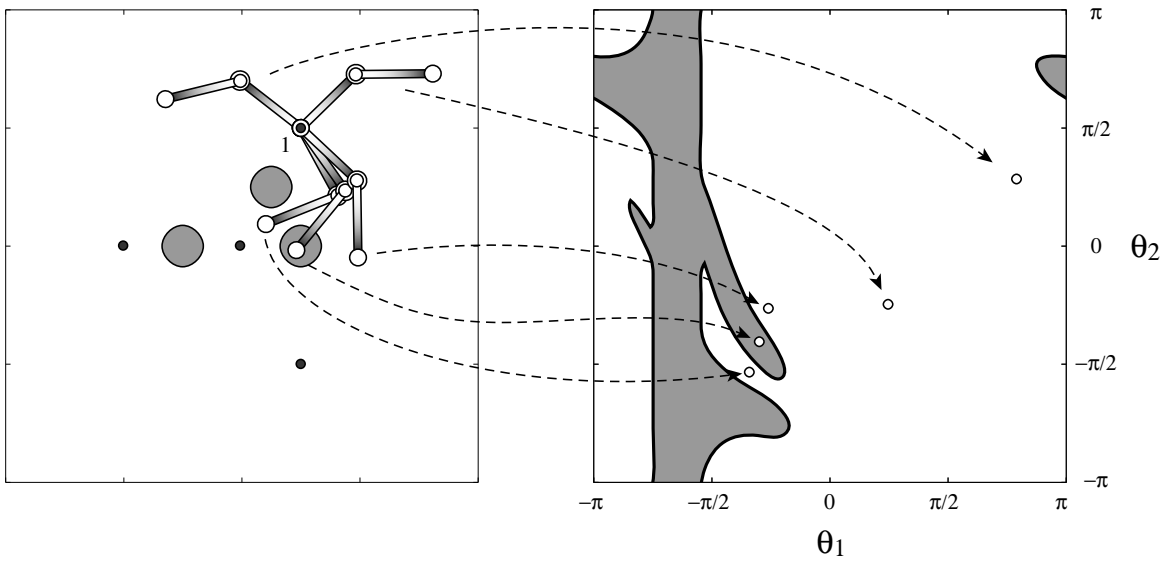


Figure 4: The 2-D feasible space F_1 (non-shaded region) associated with a stance at Hold 1. Several configurations (four feasible, one infeasible) are shown.

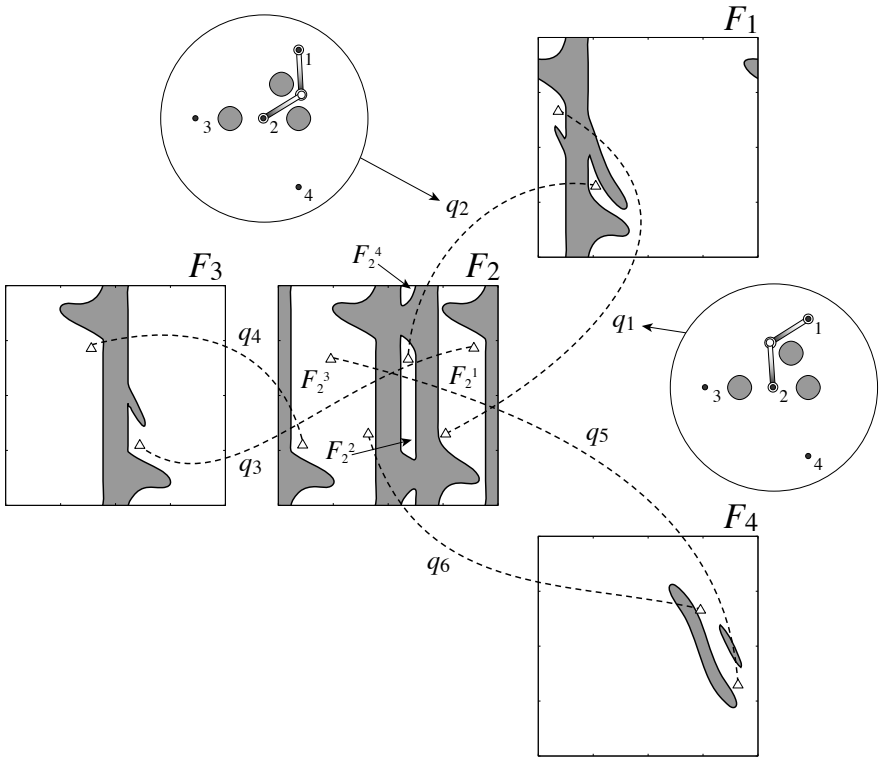


Figure 5: The transitions $q_1, \ldots, q_6$ between all four stance manifolds. Any two points connected by a dashed line represent the same configuration q_i.

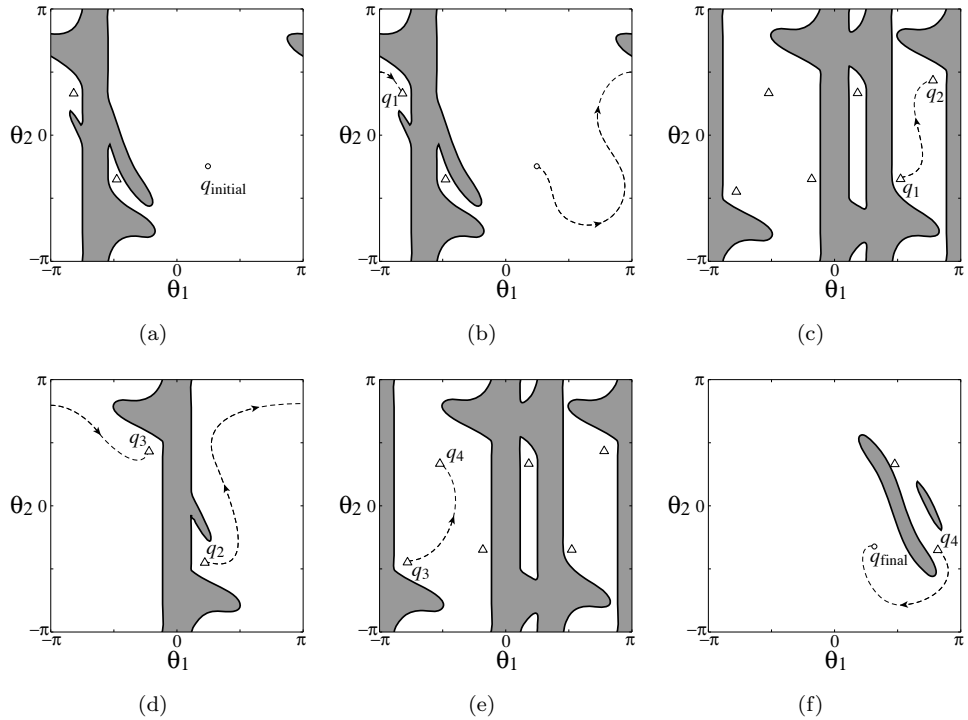


Figure 6: The feasible path corresponding to each step of Fig. 3. Angles are taken modulo 2π. Each path leads to a transition.

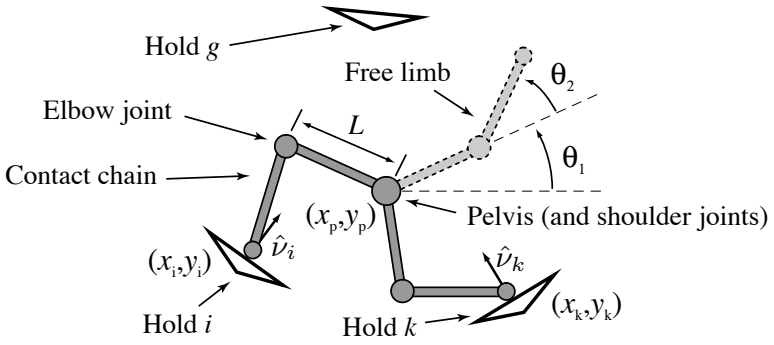


Figure 7: A three-limbed free-climbing robot on frictional point holds.

2.2 Three-limbed free-climbing robot

Next, consider the three-limbed free-climbing robot shown in Fig. 7. In addition to keeping hands at holds (like the double-link robot), this robot relies on friction to maintain equilibrium. We describe this robot in more detail because we will use it in Section 4 to develop a one-step planning algorithm.

Model and notation. The robot consists of three identical limbs meeting at a point, called the *pelvis*. Each limb has two links and two actuated, revolute joints. All six links have equal mass and length. The first joint in each limb (the *shoulder*) is located at the pelvis, while the second (the *elbow*) is located between the two links. The endpoint of each limb is called a hand. Any configuration of the robot can be defined by 8 parameters: the coordinates x_p, y_p of the pelvis and the joint angles θ_1, θ_2 of each limb. The robot's configuration space is $\mathbb{R}^2 \times (S^1)^6$.

The robot's workspace is a vertical plane containing scattered holds, with which its hands can make frictional point contact. Each hold i is defined by a point (x_i, y_i), a normal direction ν_i, and a coefficient of Coulomb friction μ_i. In figures, a hold is depicted as a triangle, but the hold itself is located at the midpoint of the long edge and is oriented along the outward normal to this edge. The robot always maintains contact with either two or three holds. In Fig. 7, for example, two hands are at holds i and k while the third limb is moving. Holds i and k are the *supporting holds*. The set $\sigma = \{i, k\}$ is a stance. The two-limbed linkage between i and k is the *contact chain* and the other limb is *free*.

At a 3-hold stance $\sigma(3)$, the robot's continuous motion (assumed to be quasi-static) takes place in a 2-D stance manifold $C_{\sigma(3)}$. The subset of $C_{\sigma(3)}$ corresponding to a fixed set of elbow bends (a fixed sign of each θ_2) can be parameterized by the position (x_p, y_p) of the pelvis. Eight copies of this coordinate system – one for each combination of elbow bends – make up a complete parameterization of $C_{\sigma(3)}$. Likewise, at a 2-hold stance $\sigma(2)$, configurations are restricted to a 4-D stance manifold $C_{\sigma(2)}$. For fixed elbow bends, $C_{\sigma(2)}$ can be parameterized by (x_p, y_p) and the angles (θ_1, θ_2) of the free limb – four copies of this coordinate system make up a complete parameterization.

Static equilibrium. For the double-link robot, the feasible space at each stance consisted of collision-free configurations. Here, it consists of configurations at which the three-limbed

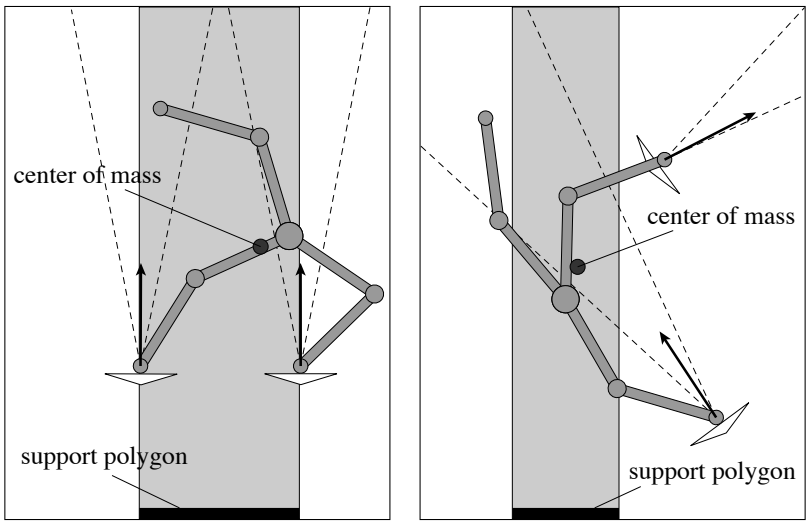


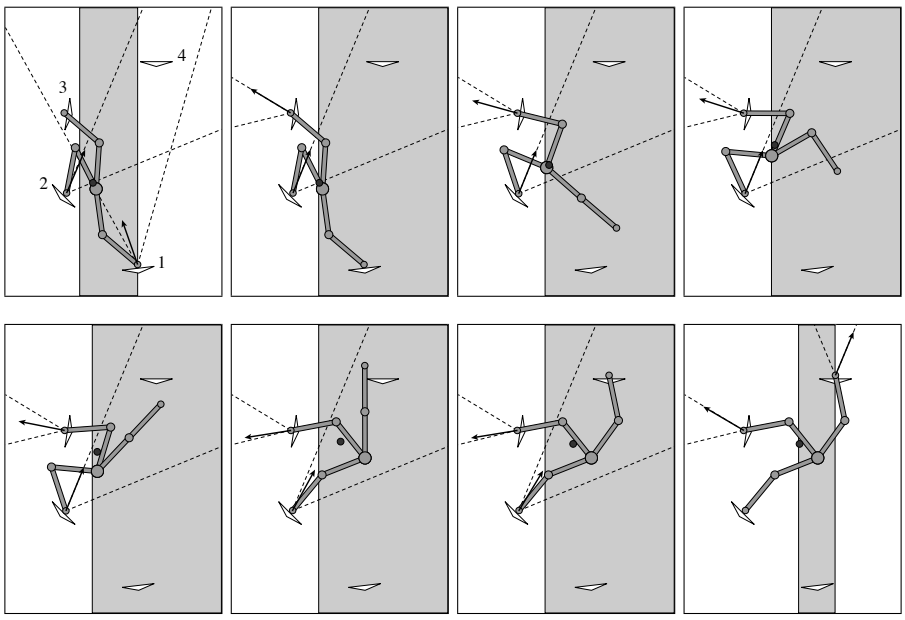
Figure 8: Balance on (a) flat and (b) uneven terrain.

robot can balance without slipping (we allow the robot's limbs to cross each other). To remain balanced, the robot must apply *contact forces* with hands at holds that compensate for gravity. To avoid slipping, these contact forces must lie inside friction cones, whose shape is governed by ν_i and μ_i at each hold. Gravity acts at the robot's center of mass (CM), the position of which varies as the robot moves. Hence, this constraint (which we refer to as *balance* or *equilibrium*) restricts the range of positions of the CM.

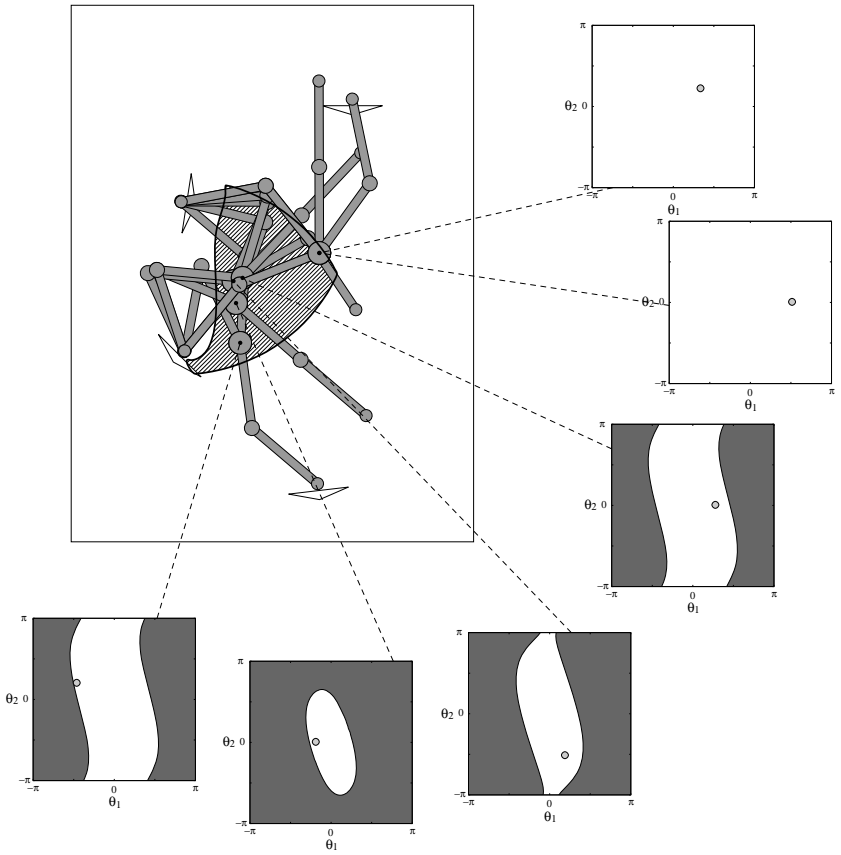
On flat horizontal terrain, a necessary condition for balance is that the robot's CM lie above the base of its supports (Fig. 8(a)). This region is called the *support polygon*, although in a planar workspace it is a segment. On uneven terrain, balance is more complex. In Fig. 8(b), not only is the robot's CM outside the base of its supports, but if it were inside, the robot would slip and fall. There is still a support polygon above which the CM must lie in order for the robot to balance, but its shape depends explicitly on the robot's stance (see Appendix A). Methods for computing the support polygon are discussed in [8, 9, 11]. Here, it is enough to know that the support polygon for the three-limbed robot is always some segment $(x_{\min}, x_{\max})$ in the workspace. The subset of configurations $F_\sigma \subset C_\sigma$ placing the CM inside the vertical column above this segment is the feasible space at σ.

If a 2-hold stance $\sigma(2)$ and a 3-hold stance $\sigma(3)$ share two holds, then they are *adjacent* and a transition $q_t \in F_{\sigma(2)} \cap F_{\sigma(3)}$ may exist between them. Here, transitions do not belong to discrete collections of points (as for the double-link robot), but rather 2-D subsets. We will occasionally use the term "transition" to refer to both the subset $F_{\sigma(2)} \cap F_{\sigma(3)}$ and a point q_t inside it – our meaning should be clear in context. A step for the three-limbed robot is a continuous path through F_σ between consecutive transitions $q_t \in F_\sigma \cap F_{\sigma'}$ and $q_{t'} \in F_\sigma \cap F_{\sigma''}$.

Example motions. A motion from $\sigma_{\text{initial}} = \{1, 2\}$ to $\sigma_{\text{final}} = \{3, 4\}$ is shown in Fig. 9(a). It consists of four steps: grab Hold 3, release Hold 1, grab Hold 4, and release Hold 2. However, q_{initial} is a transition between stances $\{1, 2\}$, $\{1, 2, 3\}$, and $\{2, 3\}$ (it lies in both $F_{12} \cap F_{123}$ and $F_{123} \cap F_{23}$), and q_{final} is a transition between stances $\{2, 3\}$, $\{2, 3, 4\}$, and



(a)



(b)

Figure 9: (a) A simple motion of the three-limbed free-climbing robot. (b) The corresponding feasible path: center is the set of feasible pelvis positions; inset are snapshots of the set of feasible free-limb angles at each pelvis position.

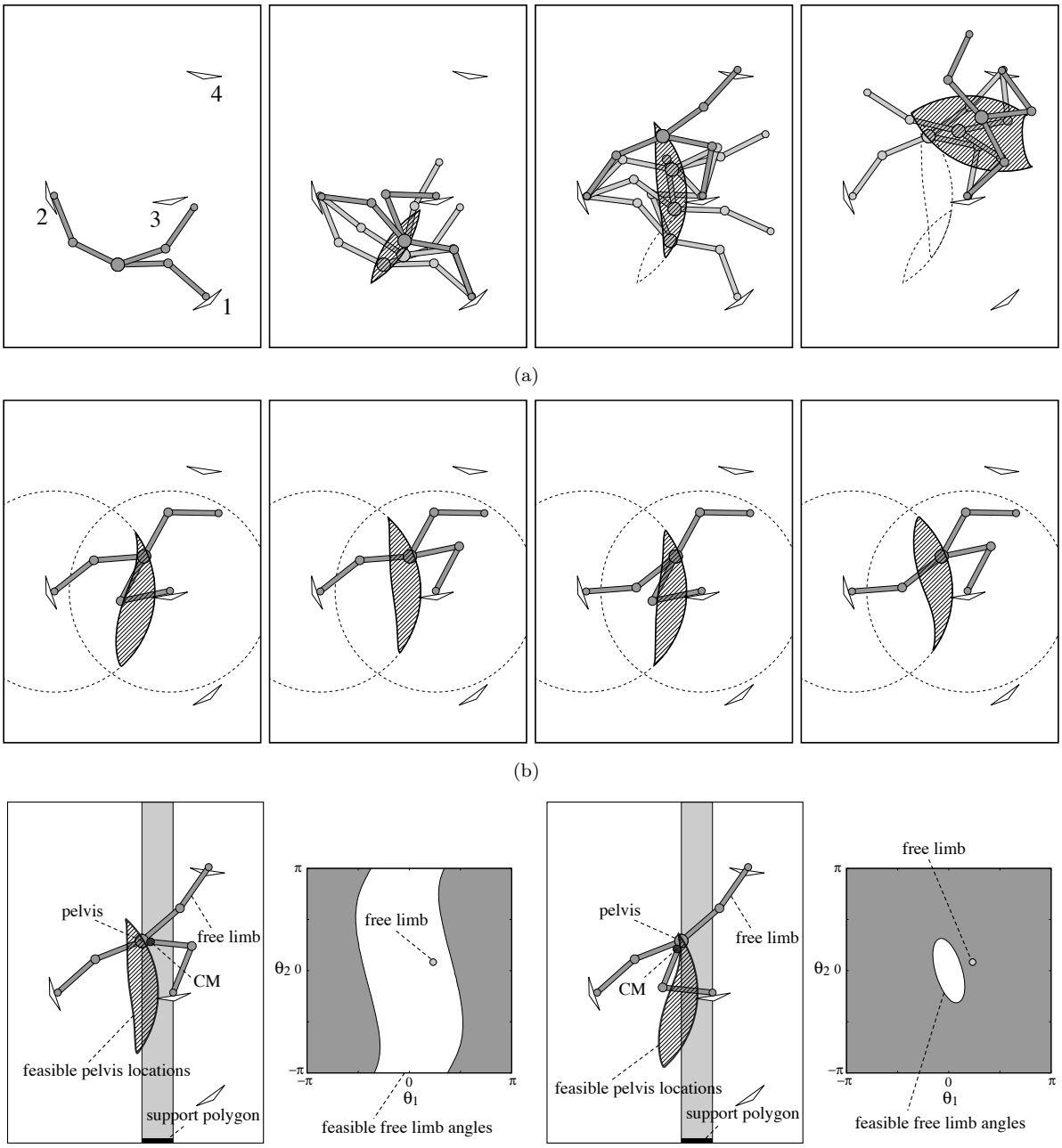


Figure 10: (a) Four steps from $\{1, 2\}$ to $\{3, 4\}$. (b) Feasible pelvis positions for each set of elbow bends at $\{2, 3\}$. It is impossible to straighten the rightmost limb without falling. (c) The transition to $\{2, 3, 4\}$ is feasible only for a certain elbow bend at $\{2, 3\}$.

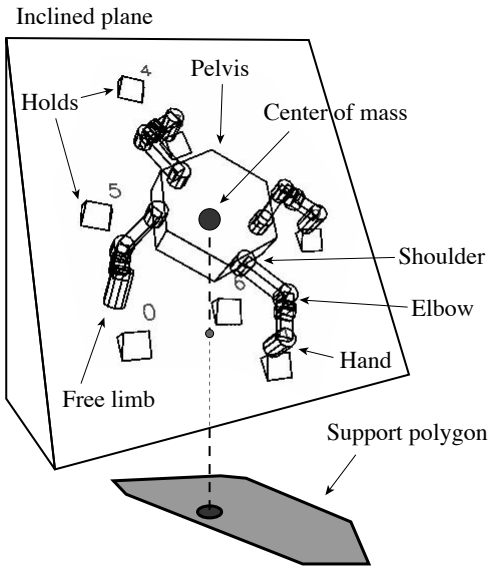


Figure 11: A four-limbed free-climbing robot (LEMUR) moving across an inclined plane containing frictional point holds.

$\{3, 4\}$ (it lies in both $F_{23} \cap F_{234}$ and $F_{234} \cap F_{34}$). So only the step to grab Hold 4, moving from q_{initial} to q_{final} in F_{23}, has nonzero length. This does not always occur, and sometimes it is necessary to move at 3-hold stances as well. Another rendering of the motion of Fig. 9(a) is displayed in Fig. 9(b). Here, the stance manifold is decomposed into two parts: one is the set of pelvis positions (cross-hatched) at which some free limb configuration places the robot in equilibrium; the other is the subset of free limb configurations that are feasible for given pelvis positions. The boundary contour of each subspace derives from the equilibrium constraint [11]. As we will see in Section 4, this decomposition is useful for planning as well as visualization.

Fig. 10(a) shows a second four-step motion. In this example, a transition from $F_{2,3}$ to $F_{2,3,4}$ is only feasible for a particular elbow bend in the limb contacting Hold 3 (Fig. 10(c)). Further, it is impossible for the robot to switch the bend in that elbow while at the stance $\{2, 3\}$ (Fig. 10(b)). Consequently, the robot must grab Hold 3 with the correct elbow bend before it releases Hold 1. Multi-step planning is necessary to make this type of decision.

2.3 Four-limbed free-climbing robot

Finally, consider a real robot, LEMUR, which we use in our hardware experiments (see Section 5). Like the three-limbed robot, LEMUR avoids falling by making frictional point contact with hands at holds. We model its (quasi-static) motion in much the same way (Fig. 11), with the following differences:

- There are four limbs instead of three, and shoulders lie along the outside of a hexagonal pelvis rather than at the center. Defining a configuration requires three additional parameters: the orientation θ_p of the pelvis and the joint angles θ_1, θ_2 of the fourth limb. Its configuration space is 11-D rather than 8-D. It switches between 3-hold and 4-

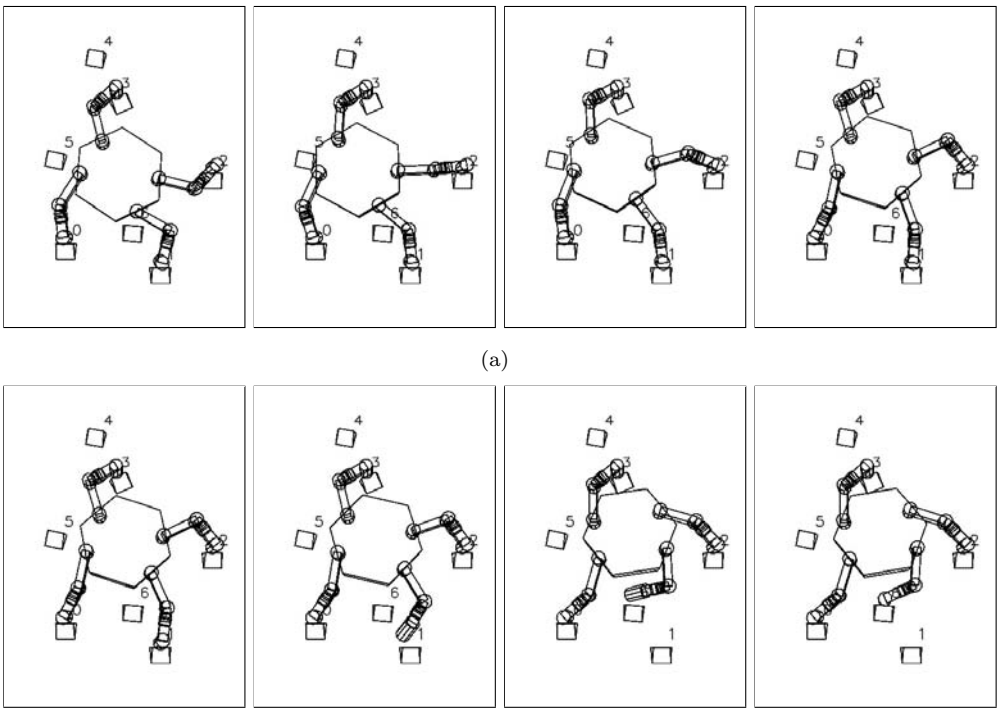


Figure 12: Two steps of a longer climb. (a) Moving to a transition to release the lower-right hold. (b) Grabbing a new hold.

hold stances (respectively 5-D and 3-D stance manifolds) rather than 2-hold and 3-hold stances.

- It climbs an inclined plane (in a 3-D workspace), rather than vertical. Holds, although located in this plane, now have coordinates and normal vectors in 3-D space. The CM moves in 3-D space, and the support polygon is an actual polygon in the plane perpendicular to gravity, not a segment (see Appendix A).
- Feasible configurations must avoid self-collision, joint-limits, and torque-limits as well as satisfy equilibrium and closed-chain kinematic constraints.

Fig. 12 shows two example steps. First (at a 4-hold stance), LEMUR changes its internal configuration to reach a transition at which it can subsequently release a hold. Second (at a 3-hold stance), LEMUR brings its free hand to a new hold. Although it is possible (but not easy) to generate a multi-step trajectory "by hand" for the three-limbed robot, this is no longer practical for LEMUR, as its stance manifolds have higher dimensionality and geometric complexity.

3 Multi-step planning framework

Every time a free-climbing robot takes a step, it faces a dilemma: it can't know the constraints on its subsequent motion until it chooses a hold, a choice it can't make until it

knows where to move next. In this section we present a framework that breaks this problem into manageable pieces. It searches the collection of stance manifolds making up the robot's configuration space (see Section 2) in two stages: first, it generates a candidate sequence of hand placements by finding transitions between manifolds; second, it refines this sequence into a feasible, continuous trajectory by finding paths between subsequent transitions.

3.1 Related work

Free-climbing is similar to other types of multi-limbed locomotion and to robotic manipulation (Section 1.2). In each case we must reason about both discrete changes in contact state and continuous motion between contact states. Previous approaches differ primarily in which part of the problem they consider first.

3.1.1 Motion before stance

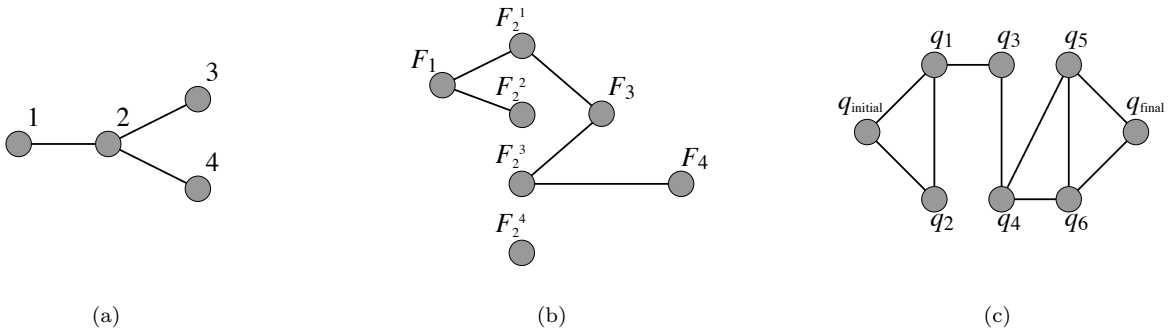
Often it may not matter much where the robot contacts its environment. For example, imagine a humanoid robot traversing nearly-horizontal terrain or an object being handled by a multi-arm robotic system. In these cases, it makes sense first to compute a robot's (or object's) overall motion, and next to derive a sequence of stances (specific footsteps or grasps) from this motion.

One technique proposed for manipulation planning is first to plan a trajectory for the grasped object ignoring manipulators, then to compute manipulator trajectories that achieve necessary re-grasps [27]. A similar approach for humanoid navigation among obstacles on flat ground is first to plan a 2-D collision-free path of a bounding cylinder, then to follow this path with a fixed gait [28, 39]. This approach has been generalized to other multi-limbed robots by using the concept of "gait controllability" to design a set of gaits that allow arbitrary collision-free paths to be followed [13, 19]. A related strategy on somewhat uneven terrain (where achieving equilibrium is still not too difficult) is first to plan a path for the CM (ignoring foot placement), then to compute specific footsteps and limb motions that keep the CM stable [18]. All of these techniques can generate multi-step motions quickly, but they do not extend well to severely uneven or broken terrain, and so are unlikely to work for free-climbing robots.

3.1.2 Stance before motion

For a free-climbing robot, the choice of contact location is critical. A few parts of the terrain may have high utility (such as a sharp horizontal ledge that can produce a wide range of reaction forces to oppose gravity), while the rest has low utility (such as a sloping bulge that can produce only a narrow range of reaction forces). In this case, it makes sense first to decide where the robot should place its hands or feet, and next to plan the continuous motion to do so.

Most previous work is based on the approach to manipulation planning proposed by [2]. This approach views (as we do) the manifold associated with each set of contacts as a separate configuration space, and expresses the connectivity among manifolds as a "manipulation graph." The challenge is to compute the structure of this graph efficiently. For specific



(a) (b) (c)

Figure 13: The (a) stance graph, (b) component graph, and (c) transition graph representing the configuration space for the double-link robot example (Fig. 5).

systems, its exact structure can sometimes be computed quickly. The work of [6] uses analytical techniques for "spider-robots" walking on horizontal terrain. For more general systems, exactly computing the graph no longer seems practical. One way to simplify the problem is to assume partial gaits. For robots in tunnels, it is possible to restrict the order in which limbs are moved [43] and to assume massless limbs [34]. For peg-climbing robots, it is possible to restrict actions to high-level modules, like "grab the nearest peg" [4]. For humanoid robots on horizontal terrain, it is possible to restrict the next footstep to a discrete set [29]. But when motion is distinctly non-gaited (as in manipulation planning [37, 41] or free-climbing), each step requires the exploration of a configuration space. Then, it becomes critical to decide how to allocate planning time over all possible steps. This problem, partially addressed by [37], motivates the two-stage search strategy we will present in Section 3.3.

3.2 Representation

A free-climbing robot's configuration space consists of a collection of stance manifolds (Section 2). We represent the connectivity among manifolds in three ways: a graph of stances, a graph of components, and a graph of transitions.

3.2.1 Double-link robot

Fig. 5 showed intersections between the 2-D manifolds F_i associated with each stance of the double-link robot. Fig. 13(a) shows the structure of these intersections as a graph. Each node is a stance. Each pair of nodes i, j is connected by an edge if there is a transition between F_i and F_j. So the robot can take a step from one stance to another only if there is an edge between the corresponding nodes in the graph. We call this representation the *stance graph*.

However, this graph does not always provide enough information to compute multi-step motions. For example, nodes 1 and 4 are connected by the edges

$$1 \rightarrow 2 \rightarrow 4.$$

But this sequence of steps is impossible (due to collision with obstacles). Instead, the robot

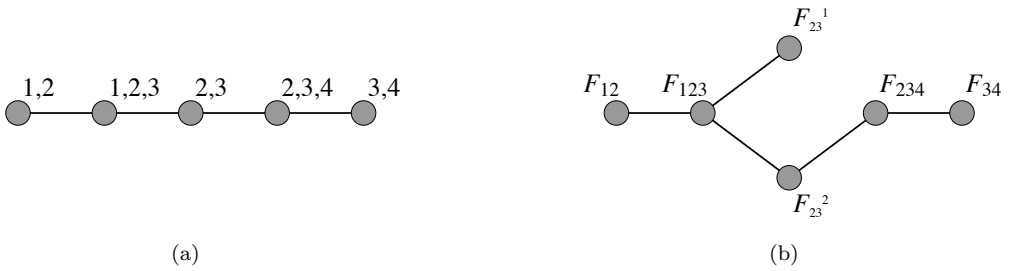


Figure 14: Graphs for the three-limbed robot examples (Figs. 9-10). (a) The stance graph (same for both examples). (b) The component graph for the second example.

must traverse the following edges (as in Figs. 3 and 6):

$$1 \to 2 \to 3 \to 2 \to 4.$$

Further, if the robot had started at a different configuration $q'_{\text{initial}} \in F_2^4$ (see Fig. 5), then it could not have taken a single step, despite the existence of edges $2 \to 1$, $2 \to 3$, and $2 \to 4$. A sequence of edges in the stance graph only provides a necessary condition for the existence of a feasible multi-step motion.

Both necessary and sufficient conditions are provided by the *component graph* (Fig. 13(b)). Each node is one connected component of a stance manifold. Two nodes F_i^h, F_j^k are connected by an edge if $F_i^h \cap F_j^k$ is nonempty. The path

$$F_1 \to F_2^1 \to F_3 \to F_2^3 \to F_4$$

in the component graph recovers the correct sequence of edges

$$1 \to 2 \to 3 \to 2 \to 4$$

in the stance graph, showing that it corresponds to a feasible multi-step motion.

Necessary and sufficient conditions can also be represented by the *transition graph* (Fig. 13(c)). Each node is a transition between manifolds (see Fig. 5): for example, $q_1 = F_1 \cap F_2^1$ and $\{q_5, q_6\} = F_2^3 \cap F_4$. Two nodes $q \in F_i \cap F_j$ and $q' \in F_j \cap F_k$ are connected by an edge if there is a continuous path between them in F_j. Fig. 13(c) is dual to Fig. 13(b), in the sense that each node (edge) of the transition graph maps to an edge (node) of the component graph, although this mapping may not be one-to-one.

3.2.2 Three-limbed robot

For the double-link robot, each stance manifold is 2-D and each transition consists of two points. For the three-limbed climbing robot, however, each manifold is either 2-D or 4-D and each transition is a 2-D submanifold that may have multiple components. So a concise representation of the configuration space becomes important. In particular, the motion in Fig. 9 involves several stances, each associated with a different support polygon and a different stance manifold. In this example each manifold only has one connected component, so the motion can be represented simply as a path through the stance graph (Fig. 14(a)). The motion in Fig. 10 is more complicated – it traverses the same stance graph, but the manifold F_{23} has two connected components (corresponding to a difference in elbow bend). Here we must look at the component graph to understand the robot's motion (Fig. 14(b)).

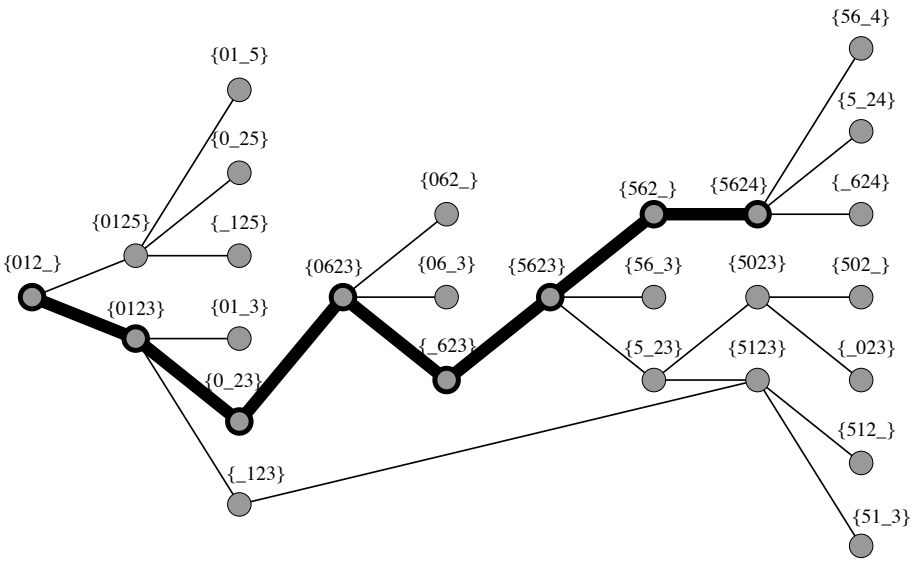


Figure 15: The stance graph for the four-limbed robot example (Fig. 12). Traversed edges are highlighted.

3.2.3 Four-limbed robot

The configuration space of the four-limbed climbing robot is significantly more complex than for the other two examples. This robot has more degrees of freedom and must avoid joint-limits, torque-limits, and self-collision in addition to staying balanced on fixed holds. In fact, the shape of each stance manifold would take a long time to compute. However, the stance graph already provides a lot of information (Fig. 15). In particular, it indicates two potential sequences of steps from $\sigma_{\text{initial}} = [012\_]$ to $\sigma_{\text{final}} = [5625]$:

$$[012\_] \to [0123] \to [0\_23] \to [0623] \to [\_623] \to [5623] \to [562\_] \to [5624]$$

and

$$[012\_] \to [0123] \to [\_123] \to [5123] \to [5\_23] \to [5623] \to [562\_] \to [5624].$$

The first of these is actually feasible (the two steps $[0123] \to [0\_23] \to [0623]$ were shown in Fig. 12), as can be verified by exploring either the component graph or the transition graph restricted to the candidate sequence.

3.3 Two-stage search

At the instant a human climber is about to grab a hold, he must be able to reach the hold but also to balance without it. In comparison, the rest of his motion (when he only has to worry about maintaining balance, or later when he can use the new hold to help him do so) is relatively easy. Consequently, he is likely to plan his route up a cliff face in two stages: first, picking out a sequence of holds to use and picturing those moments of transition between them; second, trying to decide if he can actually move from one transition to another. Here, we propose a similar strategy for free-climbing robots.

3.3.1 Algorithm

Our planner alternates between exploring the stance graph (a coarse representation of the connectivity among stance manifolds that can be generated quickly, as it only requires computing points of intersection) and the transition graph (a detailed representation that is slower to generate, as it may require computing many continuous paths). We could have used the component graph for the second stage. However, the transition graph is computationally more convenient since it consists of points and paths rather than sets and intersections. Hence, it can be constructed using sample-based techniques (Section 4).

The algorithm EXPLORE-STANCEGRAPH searches the stance graph (Fig. 16). It maintains a priority queue Q of nodes to explore. When it encounters 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. 16). It explores a transition $q \in F_\sigma \cap F_{\sigma'}$ only if (σ, σ') is an edge along the candidate sequence, and a path between $q, q' \in F_\sigma$ only if σ is a node along this sequence. We say that EXPLORE-TRANSITIONGRAPH has *reached* a stance σ_i if some transition $q \in F_{\sigma_{i-1}} \cap F_{\sigma_i}$ 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.

EXPLORE-STANCEGRAPH and EXPLORE-TRANSITIONGRAPH require two subroutines: FIND-TRANSITION, which returns a point of transition between stances σ, σ' or failure if $F_\sigma \cap F_{\sigma'}$ is empty; and FIND-PATH, which returns a path between configurations $q, q' \in F_\sigma$, or failure if q, q' are not path-connected. These subroutines are described in Section 4.

3.3.2 Why it works

The two-stage search strategy takes advantage of three observations:

- Finding a transition (a single point $q \in F_\sigma \cap F_{\sigma'}$) takes much less time than finding a path to reach it (a continuous curve in F_σ). So it is much cheaper to compute the stance graph than the transition graph.

- In practice, a feasible path often exists between two feasible transitions, since transitions are the most constrained configurations along a multi-step path. In other words, if $F_\sigma \cap F_{\sigma'}$ and $F_{\sigma'} \cap F_{\sigma''}$ are nonempty, then path-connected configurations $q \in F_\sigma \cap F_{\sigma'}$ and $q' \in F_{\sigma'} \cap F_{\sigma''}$ are likely to exist. Hence, the stance graph is usually a good approximation of the transition graph.

- Most edges in the transition graph are not on the final multi-step path. By providing necessary conditions for the existence of a multi-step path, the stance graph allows large portions of the transition graph to be pruned.

Similar observations were made by [42] about PRM planners computing collision-free paths, motivating their decision to delay costly collision checks of local paths until after finding a candidate sequence of collision-free milestones.

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

Figure 16: Algorithms to explore the stance graph and the transition graph.

3.3.3 Ordering the search

Our search strategy is a method of postponing costly computation through initial approximation (sometimes referred to as a "lazy" strategy). It can be improved by applying a heuristic to decide which approximations to refine first (sometimes referred to as a "fuzzy" strategy). We order the priority queue Q used in EXPLORE-STANCEGRAPH to make it more likely that the candidate sequence $[\sigma_1, \ldots, \sigma_n]$ is actually feasible when tested by EXPLORE-TRANSITIONGRAPH. To do this, we associate a probability $P(\sigma, \sigma')$ with every edge in the stance graph, indicating how likely it is for arbitrary configurations $q \in F_\sigma, q' \in F_\sigma \cap F_{\sigma'}$ to be path-connected. We can define this probability as inversely proportional to the amount of time spent exploring the stance manifold F_σ and the transition $F_\sigma \cap F_{\sigma'}$ [37] or based on a learned classifier [22]. We define the cost of a path $[\sigma_1, \ldots, \sigma_n]$ as

$$n - \sum_{i=1}^{n-1} \ln P(\sigma_i, \sigma_{i+1})$$

and the cost of a node σ as the minimum cost of a path from σ_{initial} to σ. Then, on Line 3 of EXPLORE-STANCEGRAPH the node σ has minimum cost, and on Line 5 we select the path of minimum cost. This heuristic also allows us to relax an implicit assumption, that if an edge (σ, σ') proves impossible along one candidate path, then this edge is impossible along any path and can be deleted. Occasionally this assumption does not hold in practice, causing EXPLORE-STANCEGRAPH to return "failure" incorrectly. So, rather than delete (σ, σ'), we reduce its probability.

4 Planning each step

The algorithms given in Section 3 rely on two capabilities: finding transitions between adjacent stances, and finding paths between subsequent transitions. Here, we present algorithms that provide these capabilities. Since a free-climbing robot may have to consider many potential steps over a long climb, it is important that these algorithms be very fast.

4.1 Probabilistic roadmaps

For a free-climbing robot, each step is different and requires the exploration of a complex, continuous configuration space. The Probabilistic-Roadmap (PRM) approach, or one of its variants, is widely used for planning paths through high-dimensional configuration spaces subject to multiple constraints (see [26] or Chap. 7 of [15]). It derives from the assumption that checking whether a configuration is feasible can often be done quickly, even if computing an exact representation of feasible space would take prohibitive time. A PRM planner samples configurations at random, retaining feasible ones as *milestones*. If possible, it connects close milestones with feasible *local paths* (often, straight-line paths). The milestones and local paths together make up the *probabilistic roadmap*. Feasible configurations are declared path-connected if they are connected in the roadmap.

Given a fixed amount of time, a PRM planner is not guaranteed to find a feasible path between two configurations whenever when exists. But usually PRM planners work well, finding a path after sampling only a few milestones. The reason is that feasible spaces encountered in practice often have favorable "visibility" properties such as *expansiveness* [25]. However, parts of a space (for example, near "narrow passages" [24]) may have less favorable visibility properties than others. To address this problem, *non-uniform sampling strategies* try to place samples in regions of feasible space having poor expected visibility. Some strategies are general: the *Gaussian strategy* samples pairs of configurations, retaining one only if it is feasible and the other is not [7]. Hence, it tends to sample more densely near the boundary of feasible space, where visibility is generally bad. Other strategies are more specific: one approach samples configurations of a robot arm where *manipulability* is low, for example at singular configurations [32]. Our sampling strategy is similar to [32], although we derive it in a different way and focus on the constraint of balance rather than collision.

4.2 Three-limbed robot

We present algorithms in this section to find transitions and paths for the three-limbed robot. They extend the PRM approach using a method of model reduction that captures the connectivity of the robot's 4-D feasible space by exploring a lower-dimensional subspace. We will apply the same ideas to the LEMUR robot.

4.2.1 Finding transitions

Each transition (an edge in the stance graph) lies in the the 2-D intersection $F_\sigma \cap F_{\sigma'}$ between feasible spaces at adjacent stances $\sigma = \{j, k\}$ and $\sigma' = \{j, k, g\}$. Rather than compute $F_\sigma \cap F_{\sigma'}$ explicitly, we use FIND-TRANSITION (Fig. 17) to sample configurations from $C_\sigma \cap C_{\sigma'}$ until a feasible one (in equilibrium) is generated. If no such configuration is found before a time limit is reached, then FIND-TRANSITION returns failure.

For a fixed triplet s of elbow bends, $C_\sigma \cap C_{\sigma'}$ can be parameterized by the pelvis position (x_p, y_p), sampled in the intersection of three discs with radius $2L$ at holds j, k, g. When the robot has hands at three holds there are eight distinct sets s, each mapping (x_p, y_p) to a different configuration q in Line 4.

4.2.2 Finding paths

Each path (an edge in the transition graph) lies in a stance manifold F_σ between subsequent transitions $q_{\text{initial}} \in F_\sigma \cap F_{\sigma'}$ and $q_{\text{final}} \in F_\sigma \cap F_{\sigma''}$. For simplicity, in this and the following section we only consider paths of nonzero length at 2-hold stances. To find such a path, the algorithm FIND-PATH (Fig. 17) uses the PRM approach. It samples configurations from C_σ, retaining samples where the robot is in equilibrium as milestones in the roadmap. For any two milestones that are sufficiently close, if the robot remains in equilibrium along the straight-line path joining them, then FIND-PATH retains this local path in the roadmap. The algorithm builds the roadmap until either the connected component containing q_{initial} also contains q_{final}, (and it returns a path between them), or a time limit is reached (and it returns failure). Smoothing techniques are used to improve the path returned by the algorithm.

For a fixed couplet s of elbow bends, C_σ can be parameterized by the pelvis position (x_p, y_p), sampled randomly in the intersection of two discs with radius $2L$ at holds j, k, and the angles (θ_1, θ_2) of the free limb, sampled randomly in the interval $[-\pi, \pi)$. When the robot has hands at two holds there are four distinct sets s, each mapping $(x_p, y_p, \theta_1, \theta_2)$ to a different configuration q in Line 22.

In particular, because two configurations with different sets of elbow bends may never be connected by a straight-line path in $(x_p, y_p, \theta_1, \theta_2)$, this parameterization implicitly decomposes C_σ into four subsets according to s. These four subsets are adjacent at two 3-D subspaces where the joint angle of one elbow in the contact chain is zero (that is, a "straight-elbow" configuration). Connections between the subsets are found by explicitly sampling configurations in these 3-D subspaces. If no sampled point is feasible, then FIND-PATH does not try to connect any two configurations with different elbow bends.

```
FIND-TRANSITION(σ = {j, k}, σ' = {j, k, g})
1  for i ← 1 to m
2        do sample (x_p, y_p) within distance 2L from holds j, k, g
3           for each triplet s of elbow bends
4                 do q ← (x_p, y_p, s)
5                    if q satisfies the equilibrium test
6                       then return q
7  return "failure"

FIND-PATH(σ = {j, k}, q_initial, q_final)
 1  M ← {q_initial, q_final}, P ← {}
 2  ▷ sample switches between elbow bends
 3  for i ← 1 to m
 4        do sample (x_p, y_p) within distance 2L from hold k
                    but at distance 2L from hold j
 5           sample (θ_1, θ_2) in [−π, π) × [−π, π)
 6           for each elbow bend s_k in the limb at hold k
 7                 do q ← (x_p, y_p, θ_1, θ_2, s_k)
 8                    if q satisfies the equilibrium test
 9                       then add q to M
10  for i ← 1 to m
11        do sample (x_p, y_p) within distance 2L from hold j
                    but at distance 2L from hold k
12           sample (θ_1, θ_2) in [−π, π) × [−π, π)
13           for each elbow bend s_j in the limb at hold j
14                 do q ← (x_p, y_p, θ_1, θ_2, s_j)
15                    if q satisfies the equilibrium test
16                       then add q to M
17  ▷ sample the feasible space at fixed elbow bends
18  for i ← 1 to n
19        do sample (x_p, y_p) within distance 2L from holds j, k
20           sample (θ_1, θ_2) in [−π, π) × [−π, π)
21           for each couplet s of elbow bends
22                 do q_1 ← (x_p, y_p, θ_1, θ_2, s)
23                    if q_1 satisfies the equilibrium test
24                       then add q_1 to M
25                            for each q_2 ∈ M within distance ε of q_1 ≠ q_2
26                                  do if the straight-line path from q_1 to q_2
                                             satisfies the equilibrium test
27                                        then add (q_1, q_2) to P
28           if q_initial and q_final are in the same component of (M, P)
29              then return a path from q_initial to q_final
30  return "failure"
```

Figure 17: Algorithms to find transitions and paths.

4.2.3 Model reduction and refined algorithm

The feasible space of the three-limbed robot has properties that can be used to make FIND-PATH faster. We have developed a method of model reduction that allows the connectivity of F_σ (parameterized by $(x_p, y_p, \theta_1, \theta_2)$) to be captured by exploring a lower-dimensional subspace (parameterized only by (x_p, y_p)). We described our method previously in [11] – for completeness, details are provided in Appendix B. The main results are as follows:

- At a fixed pelvis position (x_p, y_p) satisfying closed-chain constraints, it is easy to test whether there exists a free-limb configuration (θ_1, θ_2) placing the robot in equilibrium. If so, we call (x_p, y_p) feasible, and denote the subset of all such (θ_1, θ_2) by Θ_f.
- Either Θ_f is connected, or it has exactly two components. In the latter case, the two components differ by whether $\theta_1 > 0$ or $\theta_1 < 0$. There exist two continuous functions from the set of feasible (x_p, y_p) to Θ_f, one for which $\theta_1 > 0$ and one for which $\theta_1 < 0$.
- Hence, any feasible path of the pelvis can be lifted into F_σ.

So, we can visualize the configuration space easily in 2-D (in terms of the pelvis location), to understand better the interaction between balance and closed-chain kinematics. In particular, we can make the following observation: most of the time, connected components of F_σ differ by an elbow bend. Consequently, we can say precisely why FIND-PATH works well: it explicitly samples those parts of C_σ (at "straight-elbow" configurations) likely to have poor visibility properties. The only other disconnections encountered in practice occur when the pelvis is very close to one of the supporting holds – that is, when the elbow is almost completely bent. So, we arrive at the same conclusion of [32]: a good non-uniform sampling strategy for kinematic chains is to sample singular configurations explicitly. This result is also implied by the more formal analysis of [45].

4.3 Four-limbed robot

The following changes to FIND-PATH and FIND-TRANSITION are made for the LEMUR robot:

- LEMUR, unlike the three-limbed robot, is subject to joint limits. It is very inefficient to check this constraint after sampling a configuration. Often, less than 5% of sampled configurations satisfying the kinematic constraint also satisfy joint limits. Consequently, an alternative method is used, similar to those presented in [16, 21, 31]. This method generates a configuration sequentially, joint-by-joint, enforcing both joint limits and the kinematic constraint while sampling. At a small computational cost, it often increases the ratio of feasible samples to over 75%.
- LEMUR's complexity has thus far prohibited the analytical method of model reduction applied to the three-limbed robot. However, we still use the non-uniform sampling strategy motivated by this analysis (see Section 4.2.3). It works equally well for LEMUR.
- Testing configurations and paths for torque limits is left to post-processing. This constraint takes more time to test than others, and was rarely exceeded during hardware experiments, so postponing it speeds up planning.

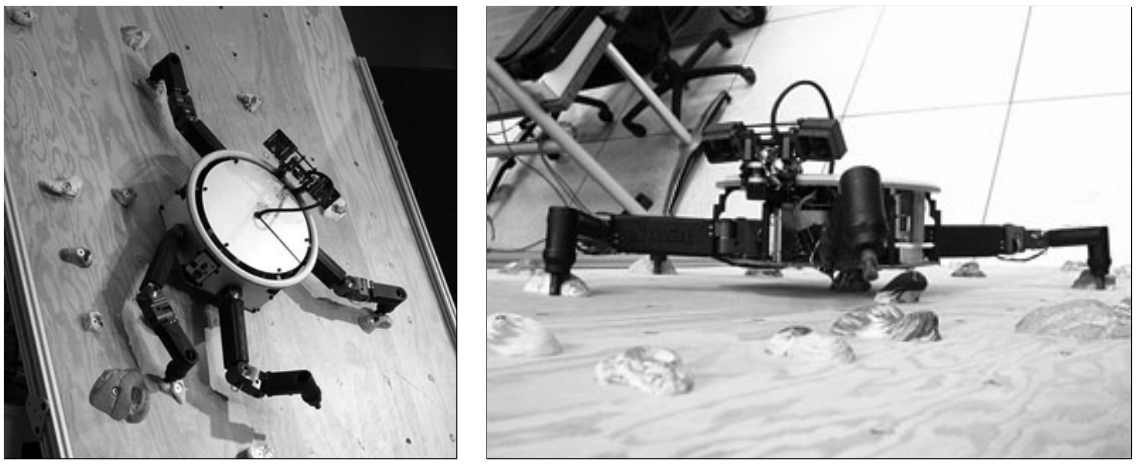


Figure 18: The free-climbing robot LEMUR, developed by NASA-JPL.

5 Implementation on a real robot

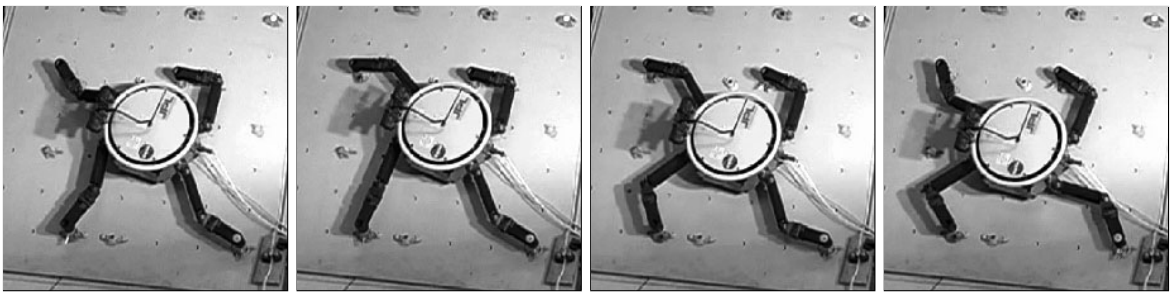
We implemented our multi-step planner on the LEMUR robot, developed by NASA-JPL. Using the planner, LEMUR can free-climb the equivalent of a human climbing gym. Of course, free-climbing also requires goal-directed sensing and reactive execution. Visual and tactile exploration is needed to search for potential holds and estimate their surface properties. Active motion control is needed to maintain balance through careful distribution of contact forces while following a trajectory. For our experiments, LEMUR has few of these capabilities (it climbs well, given these limitations). An important part of this section is to identify areas of future work based on observed failure modes.

5.1 Hardware platform

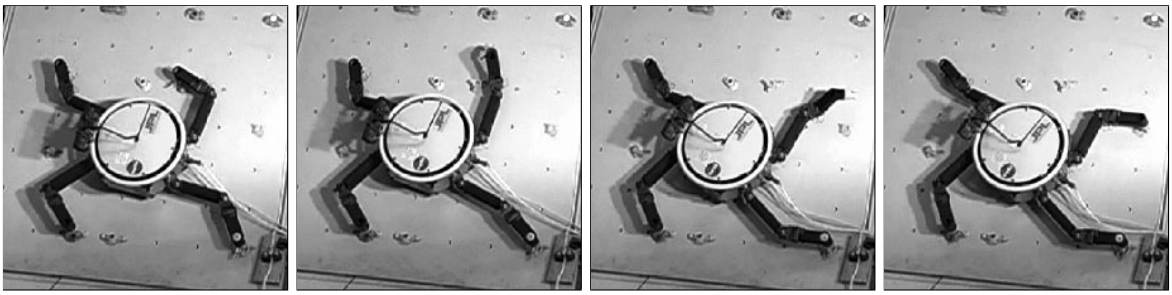
LEMUR, shown in Fig. 18, consists of four identical limbs attached to a circular chassis. The robot has a total mass of 7 kg, and a wingspan of 1 m. Each limb contains three revolute joints, providing two in-plane (yaw) and one out-of-plane (pitch) degrees of freedom. Each joint is highly geared and is capable of a maximum continuous torque of 5.0 N-m and a maximum speed of 45 deg/s. Each end-effector is a peg wrapped in high-friction rubber. The terrain climbed by LEMUR in each test is an indoor, artificial rock surface. This surface is near-vertical (varied between 50-90 degrees), planar, and covered with the same small features (holds) of irregular shape that are used in human "climbing gyms."

As input, the planner receives a presurveyed model of the terrain (a list of holds) and the robot's initial position (a configuration and a stance). A human operator specifies either a single-step goal (the next hold to grab or release) or a multi-step goal (a distant hold, requiring a number of steps to reach). As output, the planner either generates a path (a list of joint-angle waypoints and feedforward torques) to be passed to the control system, or indicates that such a path could not be found. Before the path is actually executed, it can be evaluated by the operator in a graphical user interface (GUI).

A separate proportional derivative (PD) feedback loop is used to control each of the robot's twelve motors. Joint-angle commands and feedforward torques are provided by a



(a)



(b)

Figure 19: Two consecutive steps. (a) Releasing a hold. (b) Grabbing a hold.

trajectory generator, that ramps between path waypoints at a speed inversely proportional to tracking error. By slowing down when the error increases, the control system tries to ensure that LEMUR maintains precisely coordinated limb motion and, therefore, contact with the terrain. All of the low-level electronics and software (device drivers, motor controllers, network bus) were implemented by NASA-JPL.

5.2 Experimental results

Two consecutive steps are shown in Fig. 19. In the first step, the robot adjusts its internal posture so it can subsequently release a hold without falling. In the second step, the robot brings a hand to a new hold. Both of these steps were generated autonomously by our planner, taking about one second of computation time. Indeed, they would have been difficult and slow to compute otherwise, because the robot needs to make several special adjustments to its posture in order to avoid falling. For example, in the last frame of Fig. 19(b) it is executing a classic "drop-knee" maneuver (in the language of human climbing). This relationship between what the robot is doing and what human free-climbers do is striking. Our planner does not explicitly use "human-like" maneuvers, but the motions it generates are often human-like, because human and robotic free-climbers are governed by the same set of tight motion constraints.

These two steps are part of a longer free-climb, shown in Fig. 20. Again, our planner generated the entire motion autonomously, taking several minutes of computation time. In this case, however, the robot was unable to execute the entire trajectory without failure (human operators corrected several control errors along the way). We discuss the reasons for these errors below.

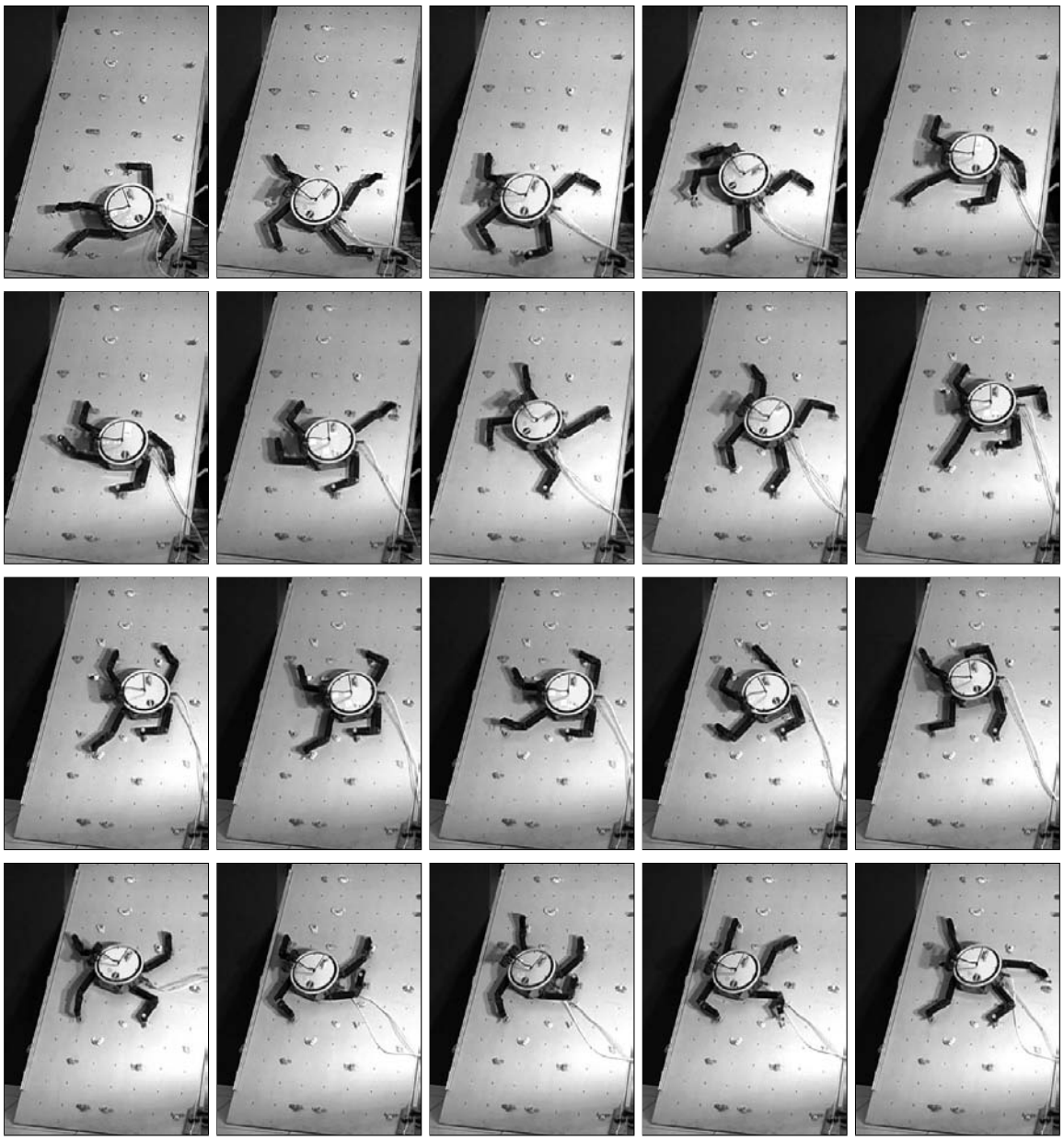


Figure 20: A long free-climb.

5.3 Lessons from experiments

LEMUR is well-designed and robust (courtesy of the team at NASA-JPL), capable of precise joint-angle control. However, several limitations remain. For example, when ascending a parallel-sided rock "chimney," a climber must not only place his hands and feet against each wall, but also continue to press outward, or he will slip. This capability is unavailable in our experiments. Instead, joint-angle control is used, achieving desired postures but not always desired forces. Likewise, in our experiments, LEMUR is unable to measure its position relative to the terrain. So, although feedback control is used to adjust internal posture, global (multi-step) motion is executed in open-loop. The lack of sensing is also why a presurveyed

model of the terrain is provided to the multi-step planner.

Given these limitations, it is remarkable that LEMUR can climb at all. Indeed, several problems arise as a direct consequence. For example, the robot occasionally slips on holds – sometimes dramatically (causing a fall), sometimes only a small amount (creating problems later on). Errors accumulate over time, eventually causing LEMUR to miss a hold that it is trying to grab. Likewise, due to the sensitivity of LEMUR's joint-angle control system, a small error in robot position and orientation may cause a large change in applied torques and forces, possibly causing the robot to exceed its torque limits.

Nevertheless, the fact that LEMUR can often climb successfully, despite these limitations, demonstrates that it is possible to make a general-purpose, multi-limbed robot free-climb, provided it is equipped with a suitable motion planner. Our hardware experiments also demonstrate the difficulty of multi-step planning. We conducted several experiments in which each individual step (the continuous motion to grab or release a hold) was planned autonomously, but the entire sequence of steps (the sequence of holds to grab or release) was specified manually by a human free-climber. The "human planner" took much longer than our planner (days rather than minutes). One reason is that it is difficult to map climbing motions across morphologies, even though the robot "climbs like a human." Also, it takes many steps for a free-climber to get anywhere, something that has to be seen to be believed. LEMUR has a wingspan of 1 meter, but to climb a distance of only 2 meters (the height of the wall) usually takes a minimum of about 80 steps, despite the arrangement of holds. It generally takes 4-12 steps for the robot even to climb to the limit of its reach.

Our experiments also indicated when a minor change in hardware design can avoid a major increase in planning complexity. For example, our planner assumes that contact points are fixed, and do not roll. However, the hands originally used by LEMUR were simple rigid pegs, with some nonzero radius. As the robot moved, its limbs rotated about holds, causing the hands to roll a short distance. Observed symptoms were similar to those caused by slip. For a free-climbing robot it is impractical to take rolling into account while planning (although this is certainly not the case in some other applications, like dexterous manipulation). Rather, this problem was corrected by a slight change in the end-effector design – they are still rigid pegs, but are offset to swivel about the axis of contact, roughly approximating an articulated wrist.

6 Conclusion

In this paper we addressed the problem of planning non-gaited motions of free-climbing robots. First, we presented a multi-step planning framework that follows a "stance before motion" approach, deciding where the robot should place its hands and feet before planning the continuous motion to do so. This two-stage strategy works well because the satisfaction of simple conditions (the existence of a transition between two manifolds) often implies that more complicated conditions (the existence of a path) are also satisfied. Next, we presented a one-step planning algorithm that extends the probabilistic roadmap approach with a particular non-uniform sampling strategy. This strategy derives from a method of model reduction that decomposes the feasible space of a three-limbed climbing robot into two subspaces (one associated with the contact-chain and one with the free-limb). Finally,

we presented an implementation for a real robot. Using our planner, LEMUR free-climbed an indoor, near-vertical surface with artificial rock features.

There are many opportunities for future work. Free-climbing is an instance of the "classic" robotics problem, ultimately requiring tight integration of sensing, planning, and control. We presented some immediate limitations of our current hardware testbed in Section 5, but we also have several longer-term objectives. For example, in this paper we assumed the robot's motion to be quasi-static. Good human climbers often exploit dynamics, either to save energy over long climbs or to advance when further progress would otherwise be impossible. A consideration of dynamics adds considerably to the dimensionality and complexity of multi-step planning. Likewise, in a field setting, a global model of the terrain will not be available in advance, so the robot will have to autonomously detect and characterize holds. Visual sensing is useful, but (unlike many other applications of computer vision) the parts of the terrain that are hardest to see (for example, pockets and cracks) are often the regions most likely to contain holds. Tactile sensing may help overcome this problem, but it would take too much time to explore the entire terrain, so the robot would have to decide where to look for holds in addition to deciding where to climb.

Free-climbing also raises interesting new computational issues. For example, a free-climb is typically characterized by a number of critical or "crux" steps that are difficult but that the robot must take, regardless of how it chooses the rest of its motion. For most search algorithms, it is hard to distinguish a critical step from an impossible one (both take a long time to explore). A fundamental question arises: how much time should be spent on each query? Too much, and time is wasted on impossible steps; too little, and critical steps are missed. We have proposed partial solutions to this problem in [10, 22]. Of course, the existence of a crux does not imply that a free-climb is hard. In general, free-climbs seem to be easy if there are either few or many holds, with hard problems in between. However, a random distribution of holds is usually either easy or unclimbable. In fact, designing hard problems is extremely difficult (human free-climbers call this process "routesetting").

Acknowledgments: This work was supported by NSF grant 0412884 and by the ATHLETE (Rough and Steep Terrain Lunar Surface Mobility) project with NASA-JPL. The author would like to thank NASA-JPL (in particular B. Kennedy and H. Aghazarian) for providing LEMUR, and J.C. Latombe, S. Rock, S. Lall, K. Hauser, and T. Miller for helpful discussions.

References

[1] M. Abderrahim, C. Balaguer, A. Giménez, J. Pastor, and V. Padrón. ROMA: A climbing robot for inspection operations. In *IEEE Int. Conf. Rob. Aut.*, Detroit, MI, 1999.

[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] M. Almonacid, R. Saltarén, R. Aracil, and O. Reinoso. Motion planning of a climbing parallel robot. *IEEE Trans. Robot. Automat.*, 19(3):485–489, 2003.

[4] 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.

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

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

[7] V. Boor, M. Overmars, and A. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Int. Conf. Rob. Aut.*, pages 1018–1023, 1999.

[8] T. Bretl. *Multi-Step Motion Planning: Application to Free-Climbing Robots*. PhD thesis, Stanford University, 2005.

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

[10] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock. Multi-step motion planning for free-climbing robots. In *WAFR*, Zeist, Netherlands, 2004.

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

[12] 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.

[13] F. Bullo and M. Žefran. Modeling and controllability for a class of hybrid mechanical systems. *IEEE Trans. Robot. Automat.*, 18(4):563–573, 2002.

[14] I.-M. Chen and S. H. Yeo. Locomotion of a two-dimensional walking-climbing robot using a closed-loop mechanism: From gait generation to navigation. *Int. J. Rob. Res.*, 22(1):21–40, 2003.

[15] 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.

[16] 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.

[17] H. Dulimarta and R. L. Tummala. Design and control of miniature climbing robots with nonholonomic constraints. In *WCICA*, Shanghai, P.R.China, 2002.

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

[19] B. Goodwine and J. Burdick. Motion planning for kinematic stratified systems with application to quasi-static legged locomotion and finger gaiting. *IEEE Trans. Automat. Contr.*, 18(2):209–222, 2002.

[20] 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.

[21] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *Int. WAFR*, 2000.

[22] K. Hauser, T. Bretl, and J.-C. Latombe. Learning-assisted multi-step planning. In *IEEE Int. Conf. Rob. Aut.*, Barcelona, Spain, 2005.

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

[24] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *WAFR*, pages 141–153, Natick, MA, 1998.

[25] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *IEEE Int. Conf. Rob. Aut.*, pages 2219–2226, 1997.

[26] 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.

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

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

[29] 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.

[30] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[31] S. M. LaValle, J. H. Yakey, and L. E. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *IEEE Int. Conf. Rob. Aut.*, Detroit, MI, 1999.

[32] P. Leven and S. Hutchinson. Using manipulability to bias sampling during the construction of probabilistic roadmaps. *IEEE Trans. Robot. Automat.*, 19(6):1020–1026, 2003.

[33] J. Long. *How to Rock Climb!* How to Rock Climb Series. Chockstone Press, May 2000.

[34] A. Madhani and S. Dubowsky. Motion planning of mobile multi-limb robotic systems subject to force and friction constraints. In *IEEE Int. Conf. Rob. Aut.*, 1992.

[35] A. Nagakubo and S. Hirose. Walking and running of the quadruped wall-climbing robot. In *IEEE Int. Conf. Rob. Aut.*, pages 1005–1012, 1994.

[36] W. Neubauer. A spider-like robot that climbs vertically in ducts or pipes. In *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, pages 1178–1185, Munich, Germany, 1994.

[37] 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.

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

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

[40] T. Roßmann and F. Pfeiffer. Control of an eight legged pipe crawling robot. In *Int. Symp. on Experimental Robotics*, pages 353–346, 1997.

[41] 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.

[42] 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.

[43] 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.

[44] M. Sitti and R. Fearing. Synthetic gecko foot-hair micro/nano-structures for future wall-climbing robots. In *IEEE Int. Conf. Rob. Aut.*, volume 1, pages 1164–1170, 2003.

[45] J. Trinkle and R. Milgram. Complete path planning for closed kinematic chains with spherical joints. *Int. J. Rob. Res.*, 21(9):773–789, 2002.

[46] M. Yim, S. Homans, and K. Roufas. Climbing with snake-robots. In *IFAC Workshop on Mobile Robot Technology*, Jejudo, Korea, 2001.

A The constraint of static equilibrium

In Section 2.3 we said that the constraint of static equilibrium restricts LEMUR's CM to lie above a support polygon. Here, we briefly derive the shape of this polygon. Consider a stance σ with N frictional point holds $r_1, \ldots, r_N \in \mathbb{R}^3$. Let $\nu_i \in \mathbb{R}^3$ 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 hold. 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, where $g = -\|g\|e_3$. Then the robot is in static equilibrium if

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

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

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

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 (that is, by linear constraints), so the set of jointly feasible contact forces and CM positions is a high-dimensional polyhedron [8, 9, 11]. Finally, since

$$c \times mg = m\|g\| \begin{bmatrix} -c_2 \\ c_1 \\ 0 \end{bmatrix}$$

then (1)-(2) do not depend on c_3, so the support polygon is the projection of this polyhedron onto the coordinates c_1, c_2. There are many ways to compute this projection and to test the membership of c. We describe an approach that works well for our application in [9].

B Model reduction and refined algorithm for the three-limbed robot

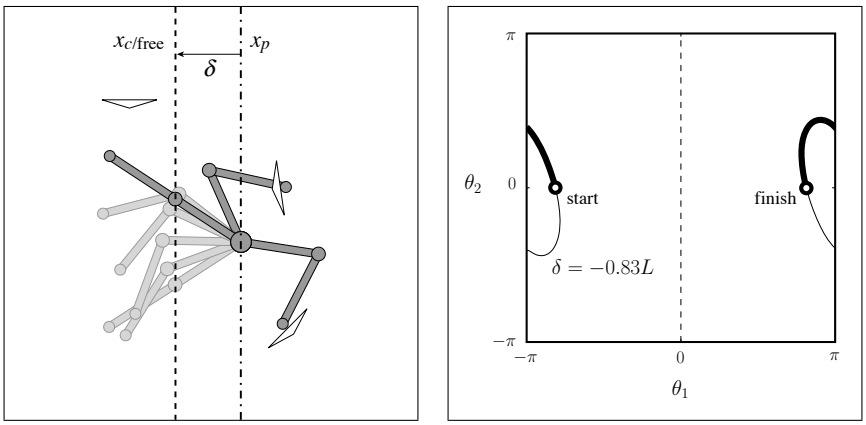
In Section 4.2.2 we presented an algorithm to find paths for the three-limbed robot. In fact, the feasible space of the three-limbed robot has properties that can be used to make this algorithm faster. In [11], we described a method of model reduction that allows the connectivity of F_σ (parameterized by $(x_p, y_p, \theta_1, \theta_2)$) to be captured by exploring a lower-dimensional subspace (parameterized only by (x_p, y_p)). We summarized our approach, and its implications, in Section 4.2.3. For completeness, we present the details here.

For a fixed configuration of the contact chain, specified by its elbow bends and the pelvis location (x_p, y_p), denote the configuration space of the free limb by $\Theta = [-\pi, \pi) \times [-\pi, \pi)$. For the robot to be in equilibrium, the abscissa x_c of its CM must remain in some interval $(x_{\min}, x_{\max})$. Since the CM of the contact chain is fixed, this constraint can be transformed into a constraint on the CM of the free limb. Let $x_{c/\text{chain}}$ and $x_{c/\text{free}}$ denote the abscissas of the CM of the contact chain and the free limb, respectively. Then $x_{c/\text{free}} = 3x_c - 2x_{c/\text{chain}}$, so $x_{c/\text{free}}$ must lie in the interval $(x_{\min/\text{free}}, x_{\max/\text{free}})$, where

$$x_{\min/\text{free}} = 3x_{\min} - 2x_{c/\text{chain}} \quad \text{and} \quad x_{\max/\text{free}} = 3x_{\max} - 2x_{c/\text{chain}}.$$

The abscissa of the CM of the free limb can be expressed as

$$x_{c/\text{free}} = x_p + \frac{L}{4}(3\cos\theta_1 + \cos(\theta_1 + \theta_2)). \tag{4}$$



(a) (b)

Figure 21: Example motion with a fixed pelvis location. (a) The motion of the robot in its workspace. (b) The path taken by (θ_1, θ_2) in Θ_f.

When θ_1 and θ_2 span Θ, the difference $\delta = x_{c/\text{free}} - x_p$ ranges between $-L$ and L. The values of θ_1 and θ_2 that are solutions of (4) for any $\delta \in [-L, L]$ define a curve in Θ of constant CM (Fig. 21). Since the mapping from (θ_1, θ_2) to Θ is single-valued, no two such curves intersect. Fig. 22(a) shows these curves for $\delta = -0.9L, -0.5L, 0, 0.5L$, and $0.9L$ for some configuration of the contact chain. Let Θ_f be the subset of Θ consisting of configurations (θ_1, θ_2) that satisfy the equilibrium test. This subset is the region between the two curves defined by $\delta_{\min} = x_{\min/\text{free}} - x_p$ and $\delta_{\max} = x_{\max/\text{free}} - x_p$. It is shown in Fig. 22(b) (shaded area), where for this example $\delta_{\min} = -0.1L$ and $\delta_{\max} = 0.7L$.

The subset Θ_f is empty if and only if $(\delta_{\min}, \delta_{\max}) \cap (-L, L)$ is empty. Given elbow bends, a configuration exists satisfying the equilibrium constraint at a fixed pelvis location if

$$\delta_{\min} < L \quad \text{and} \quad \delta_{\max} > -L. \tag{5}$$

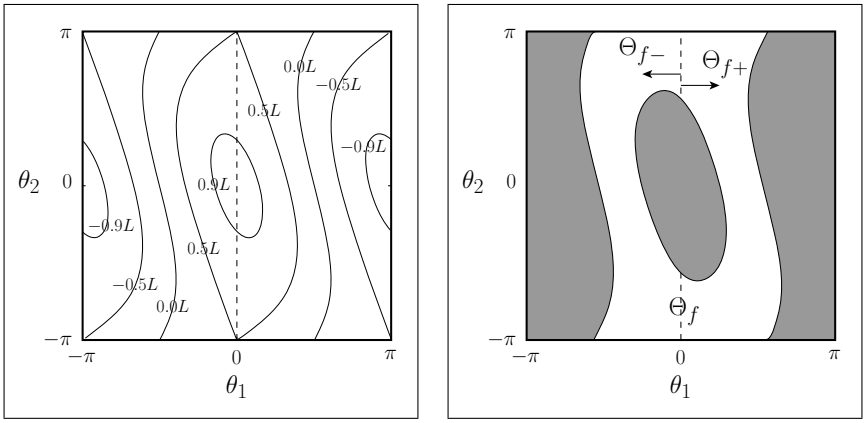
We call a pelvis location (x_p, y_p) feasible if (5) is satisfied. At any feasible pelvis location, Θ_f can be divided into two subsets: Θ_{f-}, where $\theta_1 \leq 0$ (the first link of the free limb points downward), and Θ_{f+}, where $\theta_1 \geq 0$ (the first link points upward). For any given $\delta \in (\delta_{\min}, \delta_{\max}) \cap (-L, L)$, the values of θ_1 and θ_2 that are solutions of (4) form a single continuous curve segment in Θ_{f-} and another one in Θ_{f+}. Since no two curves for distinct values of δ intersect, Θ_{f-} and Θ_{f+} are each connected. In addition, since

$$(\theta_1, \theta_2) = (-\cos^{-1}(\delta/L), 0) \quad \text{and} \quad (\theta_1, \theta_2) = (\cos^{-1}(\delta/L), 0) \tag{6}$$

are solutions of (4), they belong to Θ_{f-} and Θ_{f+}, respectively. So, each of the two segments defined by $\Theta_{f-} \cap \{\theta_2 = 0\}$ and $\Theta_{f+} \cap \{\theta_2 = 0\}$ span all feasible values of δ. Similarly, if two configurations (θ_1, θ_2) and (θ_1', θ_2') correspond to the same $\delta = \delta'$ and both belong to Θ_{f-} (respectively, Θ_{f+}), a line of constant δ joining them lies entirely in Θ_{f-} (respectively, Θ_{f+}).

It follows from the last statement that Θ_f is connected if and only if there exists θ_2 such that $(0, \theta_2)$ or $(\pm\pi, \theta_2)$ belongs to both Θ_{f-} and Θ_{f+}. This is the case if and only if at least one of the following conditions holds:

$$\delta_{\min} \notin (-L/2, L/2) \quad \text{or} \quad \delta_{\max} \notin (-L/2, L/2)\,. \tag{7}$$



(a) (b)

Figure 22: Constructing Θ_f for a fixed pelvis location. (a) Lines of constant $\delta = x_{c/\text{free}} - x_p$. (b) Example Θ_f, between $\delta_{\min} = -0.1L$ and $\delta_{\max} = 0.7L$.

Hence, any feasible continuous path of the contact chain can be lifted into F_σ by letting the free limb move in either Θ_{f-} or Θ_{f+}. This is accomplished as follows: at the start configuration of the contact chain, move the free limb in Θ_{f-} (or Θ_{f+}) from its current configuration to a configuration where $\theta_2 = 0$; next, along the path of the contact chain, continuously adjust the first joint angle θ_1 of the free limb to stay in the current Θ_{f-} (or Θ_{f+}); finally, at the end configuration of the contact chain, move the free limb in Θ_{f-} (or Θ_{f+}) to achieve a desired configuration. In some cases, it is also necessary to switch between Θ_{f-} and Θ_{f+} at configurations of the contact chain where Θ_f is connected.

This analysis yields a refined version of FIND-PATH that only samples the 2-D space of pelvis locations rather than the 4-D C_σ. Each (x_p, y_p) is lifted directly to two configurations $q_-, q_+ \in F_\sigma$ by computing the angles (θ_1, θ_2) in (6) that are guaranteed to be feasible (recall that $q_- \in \Theta_{f-}$ and $q_+ \in \Theta_{f+}$). Also, if (7) is satisfied (that is, if Θ_f is connected) then (x_p, y_p) is also lifted to a third configuration q' where the joint angle θ_1 of the free limb is either 0 or $\pm\pi$. These configurations are added to the set M of milestones. Whenever a configuration q' is generated, it is immediately connected to both q_- and q_+ in the roadmap. The purpose of adding q' is to allow changes in the sign of θ_1.

This refined version of the algorithm could very well use a grid sampling technique, since the sampled space is only two-dimensional. However, even in this case, we found that random sampling remains easily implemented and convenient.