A Foothold Selection Algorithm for Spider Robot Locomotion in Planar Tunnel Environments

Amir Shapiro
Dept. of Mechanical Engineering
Ben Gurion University, Israel

Elon Rimon[*]
Dept. of Mechanical Engineering
Technion, Israel

Shraga Shoval
Dept. of Industrial Engineering & Management
Ariel College, Israel

Abstract *This paper presents an algorithm, called* PCG, *for planning the foothold positions of spider-like robots in planar tunnels bounded by piecewise linear walls. The paper focuses on* 3*-limb robots, but the algorithm generalizes to robots with a larger number of limbs. The input to the PCG algorithm is a geometric description of the tunnel, a lower bound on the amount of friction at the contacts, as well as start and target foothold positions. Using efficient convex programming techniques, the algorithm approximates the possible foothold positions as a collection of cubes in contact c-space. Each cube represents a contact independent set of feasible* 3*-limb postures. A graph structure induced by the cubes has the property that its edges represent feasible motion between neighboring sets of* 3*-limb postures. This motion is realized by lifting one limb while the other two limbs brace the robot against the tunnel walls. A shortest-path search along the graph yields a 3-2-3 gait pattern that moves the robot from start to target using a minimum number of foothold exchanges. In practical environments the algorithm runs in time which is linear in the number of tunnel walls and polynomial in the degree of cube approximation of contact c-space. Simulations as well as experiments demonstrate the PCG algorithm in tunnel environments.*

1 Introduction

In conventional motion planning a wheeled mobile robot navigates toward a goal configuration while avoiding collision with obstacles. However, many motion planning problems are more suited for legged robots that interact with the environment in order to achieve stable locomotion. For example, surveillance of collapsed structures for survivors [18, 24], inspection and testing of complex pipe systems [19, 28], and maintenance of hazardous structures such as nuclear reactors [22] all require motion in congested, unstructured, and complex environments. Our goal is to develop general purpose multi-limb mechanisms that navigate quasistatically in such complex environments. This paper presents a polynomial time

[*]Corresponding author: Dept. of ME, Technion, Haifa, Israel 32000, elon@robby.technion.ac.il.

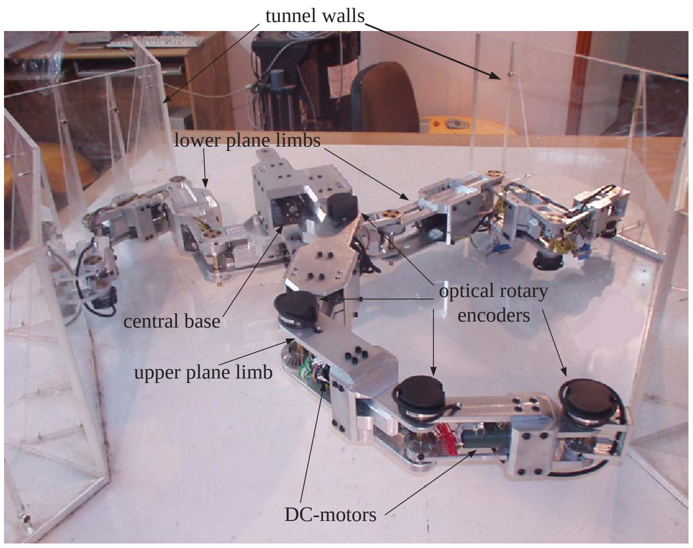


Figure 1: Top view of a 3-limb spider like robot moving in a planar tunnel environment.

algorithm, called PCG (Partitioned Cubes Gaiting), for planning the foothold positions of spider-like robots in planar tunnel environments.

A *spider-like robot* consists of k articulated limbs attached to a central body, such that each limb ends with a footpad (Figure 1). We assume that the robot moves quasistatically in a horizontal plane by exerting forces on the tunnel walls[1], while the robot is supported against gravity by frictionless contacts mounted under the mechanism. In general, a spider-like robot must have at least *three* limbs in order to move quasistatically in planar tunnel environments. At every instant the robot braces against the tunnel walls in static equilibrium using two or three limbs. During a 2-limb posture the robot moves its free limb to the next foothold position. During a 3-limb posture the robot changes its internal geometry in preparation for the next limb lifting. The PCG algorithm is presented in the context of such 3-limb robots, but the algorithm generalizes to robots having a larger number of limbs.

The foothold positions are represented as points in contact c-space (contact configuration space), which is defined as follows. Let L be the total length of the tunnel walls, and let $s_i \in [0, L]$ be an arc-length parametrization of the position of the i^{th} contact along the tunnel walls (Figure 2). Then for a k-limb mechanism *contact c-space* is the k-dimensional space $(s_1, \ldots, s_k) \in [0, L]^k$. The use of contact c-space is common in the grasp planning literature. In particular, Nguyen [17] and Ponce and his colleagues [20, 21] introduced the notion of contact independent regions. Given a k-finger grasp of a planar object, a contact independent region is a k-dimensional cube aligned with the coordinate axes in contact c-space. This cube represents k segments along the object's boundary, such that any placement of the k contacts inside these segments generates an equilibrium grasp. We use a similar notion in our representation of the feasible footholds as cubes in contact c-space. Each cube represents three segments along the tunnel walls, such that any placement of three footpads inside these segments results in a feasible 3-limb equilibrium posture. Other relevant grasping papers are those that discuss finger gaiting. Hong et al. [10] describe 3 and 4-finger gaits

[1]In quasistatic motion inertial effects due to moving parts of the robot are kept small relative to forces and torques of interaction between the robot and the environment.

for planar objects. However, they assume that once an object is grasped, the fingers may not change their order along the object's boundary. In contrast, we impose no restriction on the order of the footpads along the tunnel walls. Goodwine et al. [7, 26] investigate the stratification of the full configuration space associated with finger gaiting. While this approach is justifiable for the design of feedback control laws, motion planning can be carried out in lower dimensional spaces such as contact c-space. For example, our 3-limb robot has twelve actuated joints and three unactuated degrees of freedom at the central base (Figure 1), while its contact c-space has only three dimensions.

In the multi-legged locomotion literature, Boissonnat et al. [2, 3] discuss a motion planning algorithm for multi-legged robots in a gravitational field. They assume that the legs contact discrete point sites located on a perfectly flat terrain. Much like our approach, they lump the kinematic structure of the robot into a reachability radius, and use this parameter to design a path that takes the robot from start to target via a sequence of stable stances. Bretl, Rock, and Latombe [5] consider motion planning of a 3-limb planar robot climbing on a vertical wall under the influence of gravity. The wall contains discrete protrusions against which the robot can push during climbing. They describe an algorithm for planning a one-step motion between successive footholds, then apply a heuristic search to generate on-line a sequence of climbing steps. Our work differs from these works in several fundamental ways. First, we consider motions where the robot achieves stability by bracing against tunnel walls rather than maintaining stable stances against gravity. Second, we allow arbitrary footpad placement along the tunnel walls rather than at discrete point sites. Third, Boissonnat et al. first plan a planar path for the central body then select footholds that realize this path, while Bretl, Rock, and Latombe plan the robot motion in its full configuration space. In contrast, the PCG algorithm first plans a sequence of footholds in contact c-space, then determines the mechanism's joint values that would bring the footpads to the desired foothold positions. Other notable papers that consider multi-legged locomotion planning are [8, 9, 12, 14, 15, 25]. However, all of these papers are concerned with locomotion over a terrain in a gravitational field, while this paper is concerned with motion in congested tunnel-like environments.

This paper focuses on the portion of the PCG algorithm that plans a sequence of foothold positions in contact c-space. The algorithm consists of the following three stages. The first stage is based on a key result, that the set of feasible 3-limb postures is a union of convex sets in contact c-space. Using convex optimization techniques, the algorithm approximates each of the convex sets by p maximal cubes. In the second stage the algorithm partitions the cubes into compatible sub-cubes, where two sub-cubes are *compatible* if it is possible to move between any two postures in these sub-cubes by lifting a single limb. However, compatibility encodes only a kinematic transition between two sub-cubes. Each sub-cube is also assigned an *orientation vector* which identifies what limbs can be stably lifted from the postures in the sub-cube. The algorithm constructs a graph whose nodes are sub-cubes and whose edges connect compatible sub-cubes with suitable orientation vectors. In the third stage the algorithm searches the sub-cube graph for the shortest sequence of foothold positions that moves the robot from start to target. This sequence yields a minimal 3-2-3 gait pattern, where minimality is relative to the cube approximation of contact c-space. Application examples of the algorithm to real or simulated tunnels are shown in the simulations and experimental results sections (sections 5 and 6).

The paper is organized as follows. Section 2 characterizes the feasible 3-limb postures in contact c-space. These postures must be reachable, form stable equilibria, and satisfy

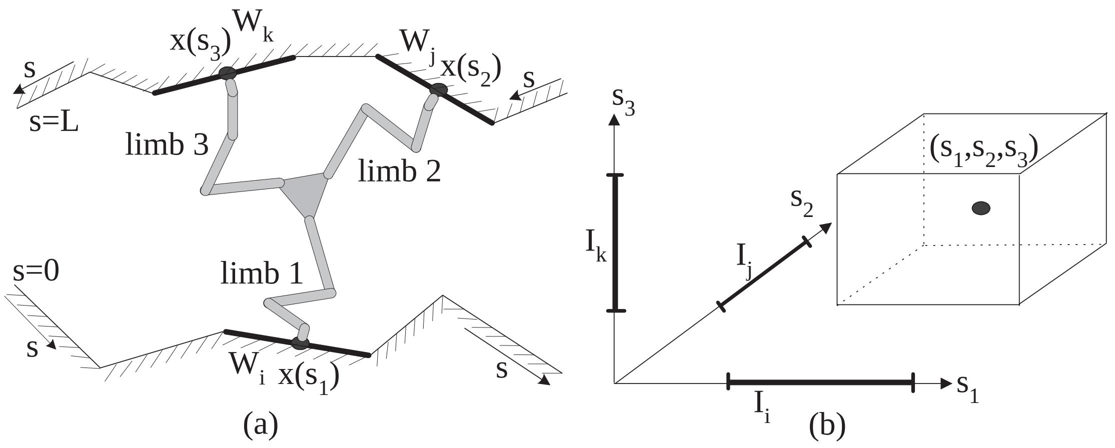


Figure 2: (a) A 3-limb robot in a planar tunnel. (b) The parametrization of its contact c-space.

a condition that allows their inclusion in a 3-2-3 gait pattern. Section 3 establishes that the feasible 3-limb postures are a union of convex sets in contact c-space. It is also shown that the approximation of a convex set by p maximal cubes is a convex optimization problem. Section 4 describes the PCG algorithm and analyzes its computational complexity. In practical tunnel environments the robot can only reach a small number of walls from any given position. In such environments the algorithm runs in $O(np^6 \log(np))$ time, where n is the number of tunnel walls and p is the number of cubes used in the approximation of contact c-space. Section 5 describes simulations of the algorithm, and Section 6 describes motion experiments with a 3-limb robot in a planar tunnel. The concluding section discusses generalization of the algorithm to cases where gravity exists and to robots having a larger number of limbs. Appendix A provides further details of the PCG algorithm. in Appendix B the effect of the parameter p on path cost is investigated and a method for its automatic selection is suggested. Finally, Appendix C contains the multimedia extensions of this paper and present a video of the robot motion in a tunnel.

2 The Feasible 3-Limb Postures

We characterize the feasible 3-limb postures as inequality constraints in contact c-space, based on the following assumptions. First, we assume piecewise linear tunnel walls with known geometry. Second, each limb contacts the tunnel walls through a footpad which can only push against the environment. Third, each footpad contacts the tunnel walls at a frictional point contact, with a known lower bound on the coefficient of friction. Fourth, the kinematic structure of the robot is lumped into a single parameter called the *robot radius* R. This parameter is the length of a fully stretched limb, measured from the center of the robot's central base. The algorithm uses this parameter to ensure that the selected footholds can be reached from the robot's central base. Recall now that contact c-space of a 3-limb robot is the cube $(s_1, s_2, s_3) \in [0, L]^3$, where L is total length of the tunnel walls and $s_i \in [0, L]$ is an arc-length parametrization of the i^{th} contact (Figure 2).

The feasible 3-limb postures must form stable equilibria, be reachable, and satisfy the following *gait feasibility* condition. This condition requires that the 3-limb posture will contain two distinct 2-limb postures—one for entering the 3-limb posture by establishing a new foothold, and one for leaving the 3-limb posture by releasing another foothold. Note that the initial and target 3-limb postures are required to contain one rather than two 2-limb postures. We now consider the individual constraints.

Equilibrium and stability of 2-limb postures. Gait feasibility requires that a 3-limb posture will contain two distinct 2-limb postures. Hence we first review the conditions for equilibrium and stability of 2-limb postures. A mechanism bracing against the environment is in *static equilibrium* if the net wrench (i.e. force and torque) generated by the contact forces acting on the mechanism as a single rigid body is zero. Once footholds are established according to the equilibrium condition, suitable feedback control laws must ensure zero net torque at each joint of the mechanism (e.g. [23, p. 98-126]). In general, a 2-limb mechanism forms an equilibrium posture as a single rigid body when the line segment connecting the two contacts lies inside the two friction cones [17]. As a stability criterion we use the notion of force closure. By definition, an equilibrium posture is *force closure* if the mechanism can resist any perturbing wrench by suitable adjustment of its contact forces with the environment [1]. In general, an equilibrium posture in a planar environment is force closure if the contact forces of the unperturbed posture lie in the interior of the respective friction cones [27].

We now write the above conditions as inequalities in contact c-space. Let $W_1, \ldots, W_n$ denote the tunnel walls, and let $I_1, \ldots, I_n$ be a partition of $[0, L]$ into intervals that represent the parametrization of the individual walls in contact c-space (Figure 2). Thus, for instance, the cube $I_i \times I_j \times I_k$ parameterizes the 3-limb postures where limb 1 contacts the wall W_i, limb 2 contacts the wall W_j, and limb 3 contacts the wall W_k. The unit tangent and unit normal to the wall W_i are denoted $\boldsymbol{t}_i$ and $\boldsymbol{n}_i$, where $\boldsymbol{n}_i$ is pointing away from the wall. Using this notation, points along W_i are given by $\boldsymbol{x}(s) = \boldsymbol{x}_i + (s - s_0^i)\boldsymbol{t}_i$, where $\boldsymbol{x}_i$ is the initial vertex of W_i, $s \in I_i$, and s_0^i is the minimal value of $s \in I_i$. Given a contact force $\boldsymbol{f}_i$, we write the force as $\boldsymbol{f}_i = f_i^t \boldsymbol{t}_i + f_i^n \boldsymbol{n}_i$, where f_i^t and f_i^n are its tangent and normal components. The Coulomb friction cone at the ith contact, denoted FC_i, is the collection of forces satisfying the inequalities: $FC_i = \{\boldsymbol{f}_i : f_i^n \geq 0 \text{ and } -\mu f_i^n \leq f_i^t \leq \mu f_i^n\}$, where μ is the coefficient of friction.

Let two limbs with indices l and m contact the tunnel walls W_i and W_j. Then for a 2-limb stable equilibrium, the vector $\boldsymbol{x}(s_m) - \boldsymbol{x}(s_l)$ must lie in the interior of the friction cone FC_i, while $\boldsymbol{x}(s_l) - \boldsymbol{x}(s_m)$ must lie in the interior of the friction cone FC_j. This condition defines a set in the (s_l, s_m) plane, denoted $\mathcal{E}_{ij}^{lm}$, which is given by

$$\mathcal{E}_{ij}^{lm} = \big\{(s_l, s_m) \in I_i \times I_j : \begin{array}{l} |(\boldsymbol{x}(s_m) - \boldsymbol{x}(s_l)) \cdot \boldsymbol{t}_i| < \mu(\boldsymbol{x}(s_m) - \boldsymbol{x}(s_l)) \cdot \boldsymbol{n}_i, \\ |(\boldsymbol{x}(s_l) - \boldsymbol{x}(s_m)) \cdot \boldsymbol{t}_j| < \mu(\boldsymbol{x}(s_l) - \boldsymbol{x}(s_m)) \cdot \boldsymbol{n}_j \big\}. \end{array}$$

An example of 2-limb stable equilibrium sets appears in Figure 7. It is important to note that the inequalities describing $\mathcal{E}_{ij}^{lm}$ are linear in s_l and s_m. Hence $\mathcal{E}_{ij}^{lm}$ is a *convex polygon* in the (s_l, s_m) plane. When $\mathcal{E}_{ij}^{lm}$ is considered as a subset of the contact c-space of a 3-limb mechanism, it becomes a three-dimensional set which is denoted as follows. Let $\Box$ serve as a place holder for the limb that does not participate in the 2-limb posture. Then a 2-limb equilibrium set, $\mathcal{E}_{ij}^{12}$ for instance, becomes a three-dimensional set which is denoted $\mathcal{P}_{ij\Box}$ and given by

$$\mathcal{P}_{ij\Box} = \big\{(s_1, s_2, s_3) \in I_i \times I_j \times [0, L] : \begin{array}{l} |(\boldsymbol{x}(s_2) - \boldsymbol{x}(s_1)) \cdot \boldsymbol{t}_i| < \mu(\boldsymbol{x}(s_2) - \boldsymbol{x}(s_1)) \cdot \boldsymbol{n}_i, \\ |(\boldsymbol{x}(s_1) - \boldsymbol{x}(s_2)) \cdot \boldsymbol{t}_j| < \mu(\boldsymbol{x}(s_1) - \boldsymbol{x}(s_2)) \cdot \boldsymbol{n}_j \big\}. \end{array}$$

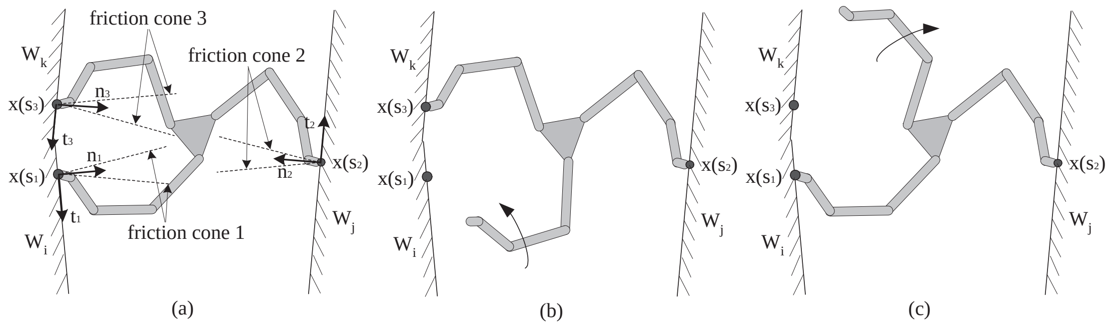


Figure 3: (a) A gait feasible 3-limb posture, (b)-(c) contains two distinct 2-limb postures.

The set $\mathcal{P}_{ij\Box}$ is a prism orthogonal to the (s_1, s_2) plane with a polygonal cross section given by $\mathcal{E}_{ij}^{12}$. Similarly, the sets $\mathcal{P}_{i\Box j}$ and $\mathcal{P}_{\Box ij}$ are prisms orthogonal to the (s_1, s_3) and (s_2, s_3) planes, with polygonal cross sections given by $\mathcal{E}_{ij}^{13}$ and $\mathcal{E}_{ij}^{23}$.

Reachability constraint of 3-limb postures. A 3-limb posture is reachable when its three footholds lie within the robot's reachability radius R. For each triplet of walls W_i, W_j, W_k, the reachability constraint is given by

$$\mathcal{R}_{ijk} = \{(s_1, s_2, s_3) \in I_i \times I_j \times I_k : \exists \boldsymbol{c} \in I\!R^2 \ \max\{\|\boldsymbol{x}(s_1) - \boldsymbol{c}\|, \|\boldsymbol{x}(s_2) - \boldsymbol{c}\|, \|x(s_3) - \boldsymbol{c}\|\} \leq R\}, \tag{1}$$

The point $\boldsymbol{c}$ appearing in (1) can be interpreted as the center of a disc containing the three foothold positions, such that the disc's radius is bounded by R. As discussed below, the elimination of the existential quantifier in (1) results in a set which is bounded by quadratic surfaces in contact c-space.

Gait feasibility of 3-limb postures. A 3-limb posture is gait feasible if it contains two distinct 2-limb equilibrium postures (Figure 3). Let us write this constraint in the contact c-space cell $I_i \times I_j \times I_k$. This cell corresponds to contact with the walls W_i, W_j, W_k, and gait feasibility is satisfied by intersection of pairs of 2-limb prisms associated with the three walls. There are three such pairs in the cell: $(\mathcal{P}_{ij\Box}, \mathcal{P}_{i\Box k})$, $(\mathcal{P}_{ij\Box}, \mathcal{P}_{\Box jk})$, and $(\mathcal{P}_{\Box jk}, \mathcal{P}_{i\Box k})$. Hence the set of all feasible 3-limb postures in the cell, denoted $\mathcal{F}_{ijk}$, is given by

$$\mathcal{F}_{ijk} = \big(\mathcal{P}_{ij\Box} \cap \mathcal{P}_{i\Box k} \cap \mathcal{R}_{ijk}\big) \cup \big(\mathcal{P}_{ij\Box} \cap \mathcal{P}_{\Box jk} \cap \mathcal{R}_{ijk}\big) \cup \big(\mathcal{P}_{\Box jk} \cap \mathcal{P}_{i\Box k} \cap \mathcal{R}_{ijk}\big). \tag{2}$$

Note that the same three walls appear in *six* cells in contact c-space, each corresponding to a specific assignment of the limbs to the three walls. The entire collection of feasible 3-limb postures is the union of all such sets over all ordered wall triplets. We end this section with an assertion that it is always possible to affect a transition between two 2-limb postures contained in a feasible 3-limb posture by suitable change of the contact forces.

Lemma 2.1. *Let a feasible* 3*-limb posture contain two* 2*-limb equilibrium postures. Then there exists a continuous change of the contact forces that allows a transition between the* 2*-limb postures, while the mechanism is kept in static equilibrium with fixed contacts.*

Proof: We present a simple continuous change of the contact forces that allows transition between the two 2-limb postures. Since the mechanism has three limbs, any two 2-limb postures must share a limb in common. Without loss of generality, let the 3-limb posture lie inside $\mathcal{P}_{ij\Box} \cap \mathcal{P}_{\Box jk}$ in contact c-space, so that limb 2 is common to both 2-limb postures. Let $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ be the contact forces at the 2-limb posture involving limbs 1 and 2, and let $\boldsymbol{g}_2$ and $\boldsymbol{g}_3$ be the contact forces at the 2-limb posture involving limbs 2 and 3. Then since the two postures are equilibrium postures we have $\boldsymbol{f}_1 + \boldsymbol{f}_2 = 0$, $\boldsymbol{g}_1 + \boldsymbol{g}_2 = 0$ and additional two equations for the torques. Multiplying the first equilibrium equation by $1-s$ and the second by s, where $s \in [0,1]$, and summing them together yields that the convex combination $(1-s)\boldsymbol{f}_1 + (1-s)\boldsymbol{f}_2 + s\boldsymbol{g}_2 + s\boldsymbol{g}_3$ generates a zero net force for all s. Applying the same methodology to the torques equations results with the very same convex combination of the forces that generates zero net wrench for all s. This convex combination specifies a continuous transition between the two 2-limb postures, while the mechanism is kept in static equilibrium. Specifically, the contact forces of limbs 1 and 3 vary only in magnitude, while the contact force of limb 2 varies in magnitude and direction between $\boldsymbol{f}_2$ and $\boldsymbol{g}_2$. Finally, since $\boldsymbol{f}_2$ and $\boldsymbol{g}_2$ lie inside the friction cone at the contact of limb 2 with the environment, their convex combination also lies inside the friction cone, for all $s \in [0,1]$. $\Box$

The lemma generalizes as follows. If a k-limb posture contains two equilibrium postures having a smaller number of limbs, it is always possible to affect a transition between these two postures by suitable change of the contact forces, while the mechanism is kept in static equilibrium.

3 Convexity of the Feasible 3-Limb Postures

In this section we discuss two issues concerning convexity that will be key to the PCG algorithm. First we establish that the feasible 3-limb postures are a union of convex sets in contact c-space. Then we show that the approximation of a convex set by p maximal cubes is a convex optimization problem.

3.1 Convexity of the feasible postures

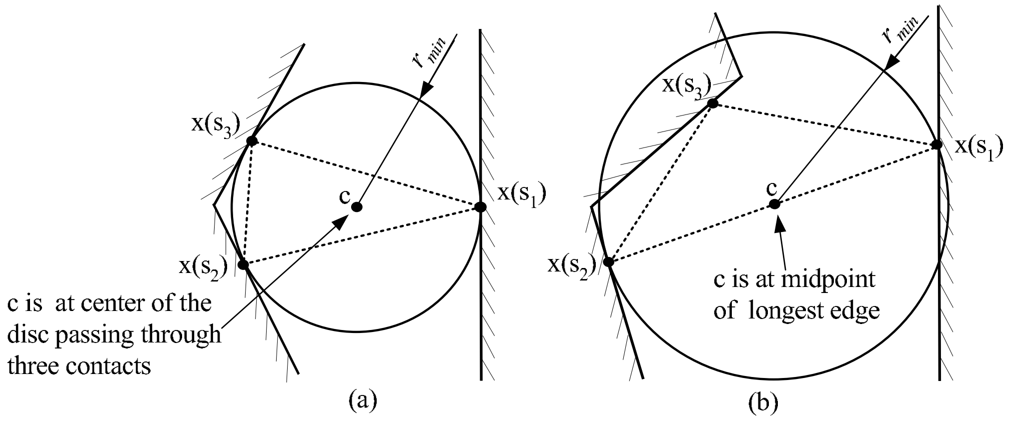
The set $\mathcal{F}_{ijk}$ of feasible 3-limb postures is specified in (2) as a union of three sets, each corresponding to a different pair of 2-limb postures. The following lemma asserts that each of these sets is convex in contact c-space.

Lemma 3.1. *In each cell $I_i \times I_j \times I_k$ of contact c-space, the set $\mathcal{F}_{ijk}$ of feasible 3-limb postures is a union of three convex sets.*

Note that any of the convex sets comprising $\mathcal{F}_{ijk}$ may be empty. For example, in Figure 8 each set $\mathcal{F}_{ijk}$ is either empty or consists of a single convex set.

Proof: The three sets that comprise $\mathcal{F}_{ijk}$ have a similar form. Hence it suffices to consider only one of these sets, say $\mathcal{P}_{ij\Box} \cap \mathcal{P}_{\Box jk} \cap \mathcal{R}_{ijk}$. The prisms $\mathcal{P}_{ij\Box}$ and $\mathcal{P}_{\Box jk}$ are defined by the intersection of linear inequalities. Each prism is therefore a convex polytope in contact c-space. Next consider the reachability set $\mathcal{R}_{ijk}$. The existential quantifier in (1) acts on a set, denoted $\bar{\mathcal{R}}_{ijk}$, which is defined in the five-dimensional space $(s_1, s_2, s_3, \boldsymbol{c})$:

$$\bar{\mathcal{R}}_{ijk} = \big\{(s_1, s_2, s_3, \boldsymbol{c}) \in I_i \times I_j \times I_k \times I\!R^2 : \max\{\|\boldsymbol{x}(s_1) - \boldsymbol{c}\|, \|\boldsymbol{x}(s_2) - \boldsymbol{c}\|, \|\boldsymbol{x}(s_3) - \boldsymbol{c}\|\} \leq R\big\}.$$



r_{min}
x(s$_3$)
c
x(s$_1$)
x(s$_2$)
c is at center of the disc passing through three contacts
(a)
r_{min}
x(s$_3$)
x(s$_1$)
c
x(s$_2$)
c is at midpoint of longest edge
(b)

Figure 4: The minimal disc containing three footholds when Δ is (a) an acute, and (b) an obtuse triangle.

The norm function $\|\boldsymbol{x} - \boldsymbol{c}\|$ is convex in $(\boldsymbol{x}, \boldsymbol{c})$ space, and each $\boldsymbol{x}(s_i)$ is linear in s_i. Since the composition of a convex function with a linear map preserves convexity, the functions $\|\boldsymbol{x}(s_i) - \boldsymbol{c}\|$ are convex in $(s_1, s_2, s_3, \boldsymbol{c})$ space. In general, the pointwise maximum of convex functions is a convex function [6, p. 47]. Hence $\bar{\mathcal{R}}_{ijk}$ is a convex set in $(s_1, s_2, s_3, \boldsymbol{c})$ space. But $\mathcal{R}_{ijk}$ is the coordinate projection of $\bar{\mathcal{R}}_{ijk}$ onto contact c-space. Since projection preserves convexity, $\mathcal{R}_{ijk}$ is convex in contact c-space. Finally, the intersection of convex sets is convex. Hence $\mathcal{P}_{ij\square} \cap \mathcal{P}_{\square jk} \cap \mathcal{R}_{ijk}$ is convex. $\square$

The PCG algorithm described below approximates the feasible 3-limb postures by cubes. The approximation requires an explicit formula for the reachable set which we now describe. The following is an equivalent formulation for $\mathcal{R}_{ijk}$,

$$\mathcal{R}_{ijk} = \big\{(s_1, s_2, s_3) \in I_i \times I_j \times I_k : r_{min}(s_1, s_2, s_3) \leq R\big\},$$

where $r_{min}(s_1, s_2, s_3)$ is the radius of the minimal disc containing the foothold positions $x(s_1)$, $x(s_2)$, and $x(s_3)$. Let Δ be the triangle generated by these three points. Then the formula for $r_{min}(s_1, s_2, s_3)$ is divided into two cases (Figure 4). When Δ is an acute triangle (i.e. with angles less than 90°), $r_{min}(s_1, s_2, s_3)$ is the radius of the disc passing through the three points, given by

$$r_{min}(s_1, s_2, s_3) = \frac{\|\boldsymbol{x}(s_1) - \boldsymbol{x}(s_2)\| \cdot \|\boldsymbol{x}(s_2) - \boldsymbol{x}(s_3)\| \cdot \|\boldsymbol{x}(s_3) - \boldsymbol{x}(s_1)\|}{2\|\boldsymbol{x}(s_1) \times \boldsymbol{x}(s_2) + \boldsymbol{x}(s_2) \times \boldsymbol{x}(s_3) + \boldsymbol{x}(s_3) \times \boldsymbol{x}(s_1)\|},$$

where $u \times v$ is the scalar obtained by taking the determinant of the 2×2 matrix $[u\ v]$. When Δ is an obtuse triangle, $r_{min}(s_1, s_2, s_3)$ is simply the half-length of the longest edge of Δ,

$$r_{min}(s_1, s_2, s_3) = \tfrac{1}{2} \max_{1 \leq p,q \leq 3} \{\|\boldsymbol{x}(s_p) - \boldsymbol{x}(s_q)\|\}.$$

The two-part formula for $r_{min}(s_1, s_2, s_3)$ reveals that the set $\mathcal{R}_{ijk}$ is bounded by quadratic surfaces in contact c-space. To summarize, the set $\mathcal{F}_{ijk}$ is the union of three *convex* sets, each bounded by planar surfaces associated with the 2-limb prisms, and quadratic surfaces associated with the reachability constraint.

3.2 Convexity of the Cube Approximation Problem

We have already established in Lemma 3.1 that the set $\mathcal{F}_{ijk}$ is a union of three convex sets. Now we discuss the approximation of these convex sets by maximal cubes. We discuss the problem in the context of three-dimensional spaces, but the result is completely general.

Consider the approximation of a three-dimensional convex set $\mathcal{S}$ by p cubes, where the cubes have an arbitrary center and dimensions. We assume as input a desired relative configuration for the cubes, where a *relative configuration* is a specification of an adjacency relation between the cubes in terms of a set of separating planes, such that no two cubes can possibly intersect. Each of the separating planes is defined only in terms of the relative position of two cubes, and does not restrict the absolute position of the two cubes. A simple example of such relative configuration is when the cubes are stacked one over the other with only one separating plane between every two adjacent cubes as shown in Figure 9. The i^{th} cube is parameterized by its center $c_i \in I\!R^3$, and its dimensions along the coordinate axes, $h_i \in I\!R^3$. The optimization therefore takes place in the $6p$-dimensional space whose coordinates are $(c_1, h_1, \ldots, c_p, h_p)$. Our objective is to maximize the total volume of the cubes, subject to constraints discussed below. However, the sum of the cubes' volumes is not a convex function of the optimization variables. Rather, we use a normalized total volume function given by[2]

$$\phi(c_1, h_1, \ldots, c_p, h_p) = \sum_{i=1}^{p} (h_{i1} h_{i2} h_{i3})^{\frac{1}{3}}.$$

Next we list the constraints involved in the cube approximation problem. First we have the requirements that the cubes' dimensions be non-negative, and that their centers lie inside contact c-space $[0, L]^3$. Second, the relative configuration of the cubes is specified by a list of separating planes, each involving the center and dimensions of two cubes separated by the plane. Last, we must ensure that the cubes lie inside the convex set $\mathcal{S}$. The following proposition asserts that the maximization of ϕ over p cubes contained in $\mathcal{S}$ is a convex optimization problem.

Proposition 3.2. *The maximization of $\phi = \sum_{i=1}^{p} (h_{i1} h_{i2} h_{i3})^{\frac{1}{3}}$ over p cubes contained in a convex set $\mathcal{S}$ and satisfying a relative-configuration specification is a convex optimization problem.*

Proof: In general, the minimization of a scalar function $\phi(x)$ subject to scalar constraints $\psi_1(x), \ldots, \psi_r(x) \le 0$ is *convex* if ϕ and $\psi_1, \ldots, \psi_r$ are convex functions of the optimization variables. In our case, the maximization of the total volume function ϕ is equivalent to the minimization of $-\phi$, and convexity of $-\phi$ is equivalent to *concavity* of ϕ. Hence we must first verify that the function $\phi = \sum_{i=1}^{p} (h_{i1} h_{i2} h_{i3})^{\frac{1}{3}}$ is concave. A sufficient condition is that the second derivative matrix of ϕ be negative semi-definite. Since ϕ depends only on the variables $h_1, \ldots, h_p$, its second derivative matrix is block diagonal, with non-zero 3×3 blocks corresponding to the second derivative of the functions $\phi_i = (h_{i1} h_{i2} h_{i3})^{1/3}$ where $i = 1, \ldots, p$. The first derivative of ϕ_i, written as a column vector, is:

$$D\phi_i = \frac{1}{3(h_{i1} h_{i2} h_{i3})^{\frac{2}{3}}} \begin{pmatrix} h_{i2} h_{i3} \\ h_{i1} h_{i3} \\ h_{i1} h_{i2} \end{pmatrix}.$$

[2]We are grateful to Prof. A. Nemirovsky who suggested this function.

The second derivative of ϕ_i is:

$$D^2\phi_i \;= \frac{1}{3(h_{i1}h_{i2}h_{i3})^{\frac{2}{3}}}\begin{bmatrix} 0 & h_{i3} & h_{i2} \\ h_{i3} & 0 & h_{i1} \\ h_{i2} & h_{i1} & 0 \end{bmatrix} - \frac{2}{9(h_{i1}h_{i2}h_{i3})^{\frac{5}{3}}}\begin{pmatrix} h_{i2}h_{i3} \\ h_{i1}h_{i3} \\ h_{i1}h_{i2} \end{pmatrix}\begin{pmatrix} h_{i2}h_{i3} \; h_{i1}h_{i3} \; h_{i1}h_{i2} \end{pmatrix}$$

Let us define the matrix H_i as follows:

$$H_i = \begin{bmatrix} -2\frac{h_{i2}h_{i3}}{h_{i1}} & h_{i3} & h_{i2} \\ h_{i3} & -2\frac{h_{i1}h_{i3}}{h_{i2}} & h_{i2} \\ h_{i2} & h_{i2} & -2\frac{h_{i1}h_{i2}}{h_{i3}} \end{bmatrix}.$$

Then we can rewrite the second derivative matrix $D^2\phi_i$ as:

$$D^2\phi_i = \frac{1}{9(h_{i1}h_{i2}h_{i3})^{\frac{2}{3}}}H_i.$$

The eigenvalues of H_i must satisfy the characteristic equation $det(\lambda I - H_i) = 0$, where λ is eigenvalue of H_i. Thus, the computed characteristic equation of H_i is:

$$\lambda\left[\lambda^2 + 2\left(\frac{h_{i1}h_{i2}}{h_{i3}} + \frac{h_{i1}h_{i3}}{h_{i2}} + \frac{h_{i2}h_{i3}}{h_{i1}}\right)\lambda + 3\left(h_{i1}^2 + h_{i2}^2 + h_{i3}^2\right)\right] = 0$$

This equation is the product of λ and a second order polynom with non-negative coefficients. One obvious eigenvalue is $\lambda = 0$. The other two eigenvalues are the roots of the second order polynom. It is well known that the roots of a second order polynom with non-negative coefficients have non-positive real part. Consequently, the matrix H_i is negative semi definite, and as a result the matrix $D^2\phi_i$ is negative semi definite. The entire matrix $D^2\phi$ is consequently negative semi-definite, and ϕ is a concave function.

Next consider the constraints on the optimization variables. First, the constraint that the cubes' dimensions be non-negative is linear in the optimization variables, and linear functions are convex. Second, a relative configuration of the cubes is specified by a list of constraints of the form: $c_{i1} + \frac{1}{2}h_{i1} \le c_{j1} - \frac{1}{2}h_{j1}$. (This particular constraint separates the i^{th} and j^{th} cubes along a plane orthogonal to the s_1-axis.) We see that the separation constraints are also linear in the optimization variables. Last consider the constraint that the cubes lie inside the convex set $\mathcal{S}$. The i^{th} cube lies inside $\mathcal{S}$ if its vertices lie in $\mathcal{S}$. We may assume that $\mathcal{S}$ is specified by inequalities $\psi_1(s_1, s_2, s_3), \ldots, \psi_r(s_1, s_2, s_3) \le 0$ such that $\psi_1, \ldots, \psi_r$ are convex functions. In this case a vertex v_j lies in $\mathcal{S}$ if it satisfies the inequalities $\psi_1(v_j), \ldots, \psi_r(v_j) \le 0$. Each vertex is given by an expression of the form $v_j = c_i \pm \frac{1}{2}h_i$ The vertices are therefore linear functions of the optimization variables. Since composition of a convex function with a linear map preserves convexity, the cube containment constraints are convex functions of the optimization variables. □

It is worth mentioning that convex optimization algorithms, for instance the ellipsoid algorithm used in our implementation, generate an ϵ-accurate solution in $O(m^2 l \log(1/\epsilon))$ time, where m is the number of optimization variables and l is the number of steps required to evaluate the constraints. An example of the approximation of a convex set by five maximal cubes using $\epsilon = 0.1$ appears in Figure 9.

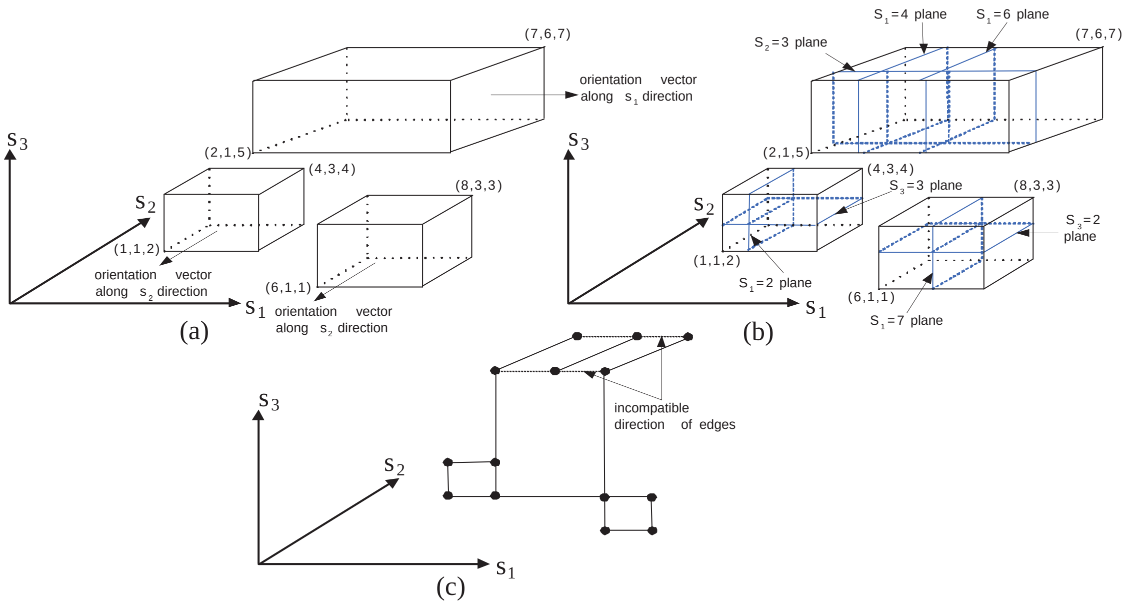


Figure 5: (a) Three cubes in contact c-space, (b) their mutual partition into sub-cubes along the separating planes, and (c) the induced sub-cube graph.

4 The PCG Algorithm

In this section we describe and analyze the PCG algorithm. First we give an overview of the algorithm. The set of feasible 3-limb postures in each cell of contact c-space is a union of three convex sets. However, in practical tunnel environments each cell contains at most one convex set. We describe the algorithm under the assumption of a single convex set per cell, and discuss the case of multiple convex sets in Appendix A. The algorithm first approximates each of the convex sets by p maximal cubes. The number of cubes and their relative configuration are user-specified inputs whose practical selection is discussed below. In order to describe the next stage of the algorithm we introduce the notion of cube orientation. A maximal cube parameterizes a set of feasible 3-limb postures, each containing two distinct 2-limb postures. The two 2-limb postures necessarily share a common limb. This common limb *cannot be lifted,* since its lifting would destroy both 2-limb postures. Each maximal cube is constructed such that its 3-limb postures have the same common limb. Hence we can associate with each maximal cube an *orientation vector,* which is aligned with the s_i-axis of the limb that cannot be lifted from the 3-limb postures parameterized by the cube. The orientation vectors play an important role in the graph construction described below.

In the second stage the algorithm partitions the maximal cubes as follows. The algorithm constructs an arrangement of all separating planes of the maximal cubes, where each separating plane contains one of the cubes' faces. Using this arrangement, the algorithm partitions the cubes as illustrated in Figure 5. The figure shows three cubes and their mutual partition along the separating planes into sub-cubes. During the partition process each sub-cube inherits the orientation vector of its parent cube. The resulting sub-cubes have disjoint interiors and satisfy the following projection property. Any two sub-cubes either have the same projection on one of the coordinate planes, or their projection on all three coordinate planes have disjoint interiors. If two sub-cubes share a projection they are called *compatible*, and the s_i-axis aligned with the direction of projection is called the *direction of compatibility.* The algorithm next defines a graph called the *sub-cube graph.* The nodes of the graph are center points of the sub-cubes. The edges of the graph connect compatible sub-cubes whose direction of compatibility is orthogonal to the orientation vector of the two sub-cubes.

Let us pause to discuss the edges of the sub-cube graph. Every edge represents lifting and repositioning of a particular limb. The lifting of a limb must leave the robot in a stable 2-limb posture. The orientation vector of a sub-cube describes which limb may not be lifted from the 3-limb postures parameterized by the sub-cube. Hence all edges emanating from a node must be *orthogonal* to the orientation vector of the sub-cube associated with the node. Moreover, all edges of the sub-cube graph are *straight lines parallel to the s_i-axes* in contact c-space (Figure 5 (c)). For example, when an edge is parallel to the s_1-axis, motion along this edge means that only limb 1 is moving, while the footholds positions of limbs 2 and 3 remain fixed. According to Lemma A.1 in Appendix A, the motion of a limb between any two sub-cubes connected by an edge can be executed such that reachability is maintained throughout the limb's motion. Finally, the start and target 3-limb postures, denoted S and T, are added as special nodes to the sub-cube graph. The construction of edges from S and T to the other nodes of the graph is described below.

In the third stage the algorithm assigns unit weight to all edges, then searches the sub-cube graph for the shortest path from S to T. The shortest path on the graph minimizes the number of limb lift-and-reposition steps from start to target. However, this minimality is only relative to the cube approximation obtained in the first stage of the algorithm. A formal description of the algorithm follows.

PCG Algorithm:

```
Input: Geometric description of $n$-wall tunnel. Coefficient of friction. Start and target 3-limb
postures $S$ and $T$. A value for number of maximal cubes $p$ and their relative configuration.
1. Cube approximation:
  1.1 Determine which cells $I_i \times I_j \times I_k$ contain a non-empty set $\mathcal{F}_{ijk}$ of feasible 3-limb postures.
  1.2 Approximate each non-empty set $\mathcal{F}_{ijk}$ by $p$ maximal cubes. Assign an orientation vector
      to each maximal cube.
2. Cube partition:
  2.1 Construct an arrangement of separating planes for all maximal cubes.
  2.2 Subdivide each maximal cube into sub-cubes along separating planes. Assign to each
      sub-cube the orientation vector of its parent maximal cube.
3. Sub-Cube Graph:
  3.1 Define a graph with nodes at center of sub-cubes and edges between compatible sub-
      cubes whose direction of compatibility is orthogonal to the orientation vector of both
      sub-cubes.
  3.2 Define $S$ and $T$ as special nodes and connect them to the graph as described below.
4. Graph search:
  4.1 Assign unit weight to all edges.
  4.2 Search for the minimum cost path from $S$ to $T$ along the sub-cube graph.
```

Two technical details of the algorithm need clarification. The first is the construction of edges from S and T to the other nodes of the graph. For simplicity, let the start and target be feasible 3-limb postures with their own orientation vector. (In general, S and T are required to contain only one stable 2-limb posture.) If the start or target lies in a sub-cube, the node associated with this sub-cube becomes the start or target node in the graph. If the start or target lies outside the cube approximation, it is added as a special node to the graph. In this case compatibility of S or T with a sub-cube means that its projection on one of the coordinate planes lies in the projection of the sub-cube. Having defined orientation and compatibility for S and T, the edges connecting these nodes to the other nodes of the graph are constructed by the rule specified in step 3.1 of the algorithm. The second technical issue is the selection of a relative configuration for the p maximal cubes. In principal any relative configuration can be used by the algorithm. In the next section we specify for each cell a relative configuration that separates the p cubes by planes having normal parallel to the cell's orientation vector. This relative configuration tends to preserve the connectivity of the convex sets in their cube approximation.

Next we discuss some notable features of the algorithm. First, the uniform edge weight assignment reflects our desire to minimize the total number of limb reposition steps along the path. However, the edges can be assigned different weights, for instance, ones that reflect a measure of distance traversed between successive footholds. Second, the sequence of footholds generated by the algorithm is contact independent in two ways. Each node of the graph parameterizes three contact independent wall segments, and each edge of the graph can be realized by limb motion between any compatible points in the sub-cubes joined by the edge. This robustness with respect to small footpad placement errors allows implementation of the algorithm using inexpensive sensors and controllers as discussed below. Third, the algorithm treats the motion of a limb between walls and along a single wall in a uniform manner. One implication of this uniformity is that small changes in the tunnel geometry, for instance a change of a long straight wall into a piecewise linear wall, would not have a significant influence on the path generated by the algorithm. Last, the size of the sub-cube graph increases with p. However, if an edge exists in the graph for low values of p, it would persist in the graph for larger values of p. Consequently, the path from start to target only becomes shorter as p increases. This issue is further discussed in Appendix B, where we suggest a method for selecting p in a way that preserves the connectivity of the sets of feasible 3-limb postures.

We now establish the correctness of the algorithm. Recall that paths in contact c-space must be orthogonal to the sub-cube orientation vectors. This constraint requires a careful consideration of the collection of reachable postures[3]. Let $\mathcal{X}$ be an approximation of the feasible 3-limb postures by maximal cubes. Let $\mathcal{R}(S)$ denote the set of 3-limb postures that can be reached from a posture S along a path in $\mathcal{X}$. The following lemma gives a sufficient condition for reachability of a target posture T.

Lemma 4.1. *Let S and T be start and target 3-limb postures. A sufficient condition for $T \in \mathcal{R}(S)$ is that the sub-cube graph contains a path from S to T whose nodes change orientation at least once along the path.*
The sufficient condition is also necessary when T lies outside the planes passing through S and orthogonal to the axes of contact c-space.

[3]The orientation vectors impose a non-holonomic constraint which has been first observed by Koditschek in the context of manipulation planning [11].

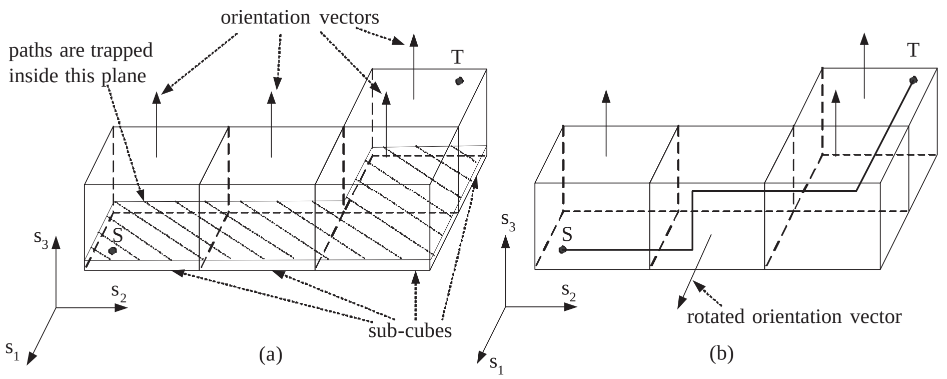


Figure 6: Four sub-cubes in contact c-space with parallel and non-parallel orientation vectors.

A proof of the lemma appears in Appendix A. The necessary condition is depicted in Figure 6(a). The figure shows four sub-cubes whose orientation vectors are aligned with the s_3-axis. The path on the sub-cube graph from S to T contains no change of orientation, and $\mathcal{R}(S)$ is trapped in a plane passing through S and orthogonal to the s_3-axis. Figure 6(b) shows the situation when one sub-cube has an orientation vector along the s_1-axis. In this case $\mathcal{R}(S)$ fills the entire sub-cube having the rotated orientation vector, as well as the subsequent sub-cubes along the path. The change-of-orientation condition of the lemma can be identified by the following simple criterion. If the sequence of footholds from S to T requires repositioning of all three limbs, the sequence contains at least one change of orientation as required by the lemma. This criterion automatically holds true when the target footholds are distinct from the start footholds.

The remainder of this section discusses the computational complexity and optimality of the algorithm. We assume in the analysis that the number of walls the robot can reach while contacting a given wall is bounded by a constant. This assumption is called *wall reachability.*

Theorem 1. *Let S and T be start and target 3-limb postures in a tunnel environment satisfying the wall reachability assumption. If T lies in $\mathcal{R}(S)$, the PCG algorithm finds a path from S to T in $O(np^6 \log(np))$ time using $O(np^6)$ space, where n is the number of tunnel walls and p is the number of maximal cubes in each non-empty cell of contact c-space.*

Proof: First consider step 1.1 of the algorithm, identifying which cells of contact c-space contain feasible 3-limb postures. The algorithm first identifies which cells contain reachable 3-limb postures as follows. The radius-R neighborhood about a wall is bounded by two linear and two quadratic curves. The collection of these neighborhoods forms a planar arrangement of $O(n)$ curves. By the wall reachability assumption, a radius-R neighborhood intersects a constant number of other radius-R neighborhoods. Hence the arrangement of radius-R neighborhoods contains $O(n)$ intersection points. A line sweep algorithm can compute the intersection points in $O(n \log(n))$ time. Each intersection point is associated with a finite number of overlapping radius-R neighborhoods, and any triplet of overlapping neighborhoods is a potentially non-empty cell in contact c-space. Thus we obtain $O(n)$ potentially non-empty cells in $O(n \log(n))$ time. The actual verification that these cells are non-empty is carried out in the next step of the algorithm.

Next consider step 1.2, where each non-empty set $\mathcal{F}_{ijk}$ is approximated by p maximal cubes. Any convex optimization algorithm first computes an initial feasible solution, or reports that no feasible solution exists. This first step determines which of the $O(n)$ cells generated by the line sweep algorithm contains a non-empty set of feasible 3-limb postures. Standard convex optimization algorithms, for instance the ellipsoid algorithm used in our implementation, generate an ϵ-accurate solution in $O(m^2 l \log(1/\epsilon))$ time, where m is the number of optimization variables and l the number of steps required to evaluate the constraints [16]. In our case $m = 6p$ since each cube has six parameters. The l constraints are the validity of the cubes' relative configuration, and the containment of the cubes' vertices in $\mathcal{F}_{ijk}$. The relative configuration consists of $p-1$ separating planes, and the total number of cube vertices is $8p$. Thus $m = O(p)$ and $l = O(p)$. The approximation of each set $\mathcal{F}_{ijk}$ by p maximal cubes takes $O(p^3 \log(1/\epsilon))$ time. Since there are $O(n)$ potentially non-empty sets $\mathcal{F}_{ijk}$, step 1.2 generates $O(np)$ maximal cubes in $O(np^3 \log(1/\epsilon))$ time.

Next consider steps 2.1 and 2.2, where the maximal cubes are partitioned into sub-cubes. Since there are $O(np)$ maximal cubes, sorting the cubes' separating planes and generating their arrangement takes $O(np \log(np))$ time. Consider now the partitioning of a maximal cube along the separating planes. By the wall reachability assumption, each maximal cube has an overlapping projection with a constant number of non-empty cells. Since a non-empty cell contains p maximal cubes, each maximal cube is partitioned by $O(p)$ separating planes along each axis of contact c-space. If we first partition the maximal cube along the s_1-axis, it is divided into $O(p)$ slabs orthogonal to the s_1-axis. The slabs are next divided along the s_2-axis into $O(p^2)$ rectangular prisms. Finally, the prisms are divided along the s_3-axis into $O(p^3)$ sub-cubes. Since there are $O(np)$ maximal cubes, step 2.2 generates a total of $O(np^4)$ sub-cubes in $O(np^4)$ time.

Step 3 concerns the construction of edges between sub-cubes. In our implementation the edges are constructed during the cube partitioning process. For purposes of analysis, let us assume that the construction of an edge takes $O(1)$ time, so that the time for step 2.3 is equal to the total number of edges in the sub-cube graph. Recall that all edges are aligned with the s_i-axes, and that an edge connects sub-cubes with a matching projection on one of the coordinate planes. Consider now a particular sub-cube denoted $\mathcal{D}$. The wall reachability assumption implies that $\mathcal{D}$ has an overlapping projection with $O(p)$ maximal cubes. Each of these maximal cubes contains a rectangular prism that has a matching projection with $\mathcal{D}$. Since a rectangular prism contains $O(p)$ sub-cubes, $\mathcal{D}$ has a matching projection with $O(p^2)$ sub-cubes. Since the sub-cubes are the nodes of the sub-cube graph, the edge degree of each node is $O(p^2)$. Since there are $O(np^4)$ sub-cubes, the total number of edges is $O(np^6)$. Note that the size of the sub-cube graph, $O(np^6)$, is the space requirement of the algorithm.

Finally consider step 4. In general, a shortest path search on a graph with m vertices and e edges takes $O(e \log(m))$ time. Substituting $m = O(np^4)$ and $e = O(np^6)$, the search for the shortest path along the sub-cube graph takes $O(np^6 \log(np))$ time. Summarizing all the steps, we obtain a run time of $O(n \log(n) + np^3 \log(1/\epsilon) + np^4 + np^6 \log(np)) = O(np^6 \log(np))$, where we made the reasonable assumption that $\log(1/\epsilon) << p^3 \log(np)$. □

Note that in practice the time required to reposition a limb is much longer than the time required to execute a computation step in the robot's computer. Hence the algorithm's relatively high run time is compensated by its ability to minimize the number of steps along the path. The following proposition characterizes this path optimality.

Proposition 4.2. *Let $\mathcal{X}$ be an approximation of the feasible 3-limb postures by maximal cubes. Let S and T be start and target 3-limb postures such that $T \in \mathcal{R}(S)$. Then up to one*

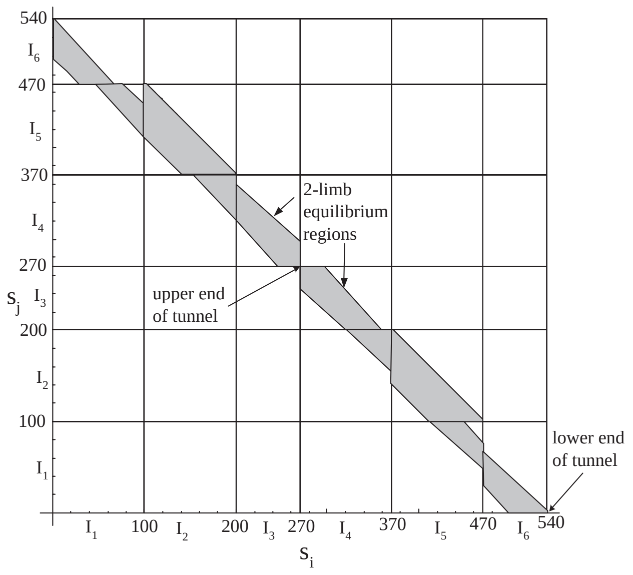


Figure 7: The 2-limb equilibrium postures in the (s_i, s_j) plane.

extra step, the path computed by the PCG algorithm minimizes the number of steps over all paths from S to T in $\mathcal{X}$.

The proof, which appears in Appendix A, realizes the path computed by the algorithm as a rectilinear path in contact c-space having one segment per edge of the sub-cube graph. Optimality of the resulting path follows from the fact that all paths associated with a 3-2-3 gait naturally embed in the sub-cube graph. Note that the sub-cube graph is a fixed data structure for a given tunnel environment. Each new start and target need only be connected to the existing graph which is then searched for the optimal path.

5 Simulation of the PCG Algorithm

This section contains results of running the PCG algorithm on a simulated tunnel depicted in Figure 12. The tunnel consists of six walls whose lengths are marked in the figure. All length units are centimeters. The figure also shows a 3-limb robot at its start and target positions. In this simulation we set the robot reachability radius to be $R = 60$ cm. The coefficient of friction is $\mu = 0.5$, a value that corresponds to rubber coated footpads contacting walls made of metal or perspex. Note that the simple tunnel already contains significant geometric features. The two walls at the tunnel's bottom form a closing cone. The tunnel next turns leftward and becomes two parallel walls. Finally, the two walls at the tunnel's top form an opening cone. These geometric features are significant, since the robot must use friction effects in order to traverse such features.

The walls are parameterized by path length in counterclockwise order (Figure 12). Thus $s = 0$ and $s = 270$ correspond to the bottom and top of the tunnel's right side, while $s = 270$ and $s = 540$ correspond to the top and bottom of the tunnel's left side. Using this parametrization, contact c-space is the cube $[0, 540]{\times}[0, 540]{\times}[0, 540]$ depicted in Figure 8. The center point of contact c-space at $(270, 270, 270)$ represents 3-limb postures where the three footpads touch the upper point of either side of the tunnel. Topologically, we ought to put a cube-shaped puncture at the center of contact c-space, since the top points on the

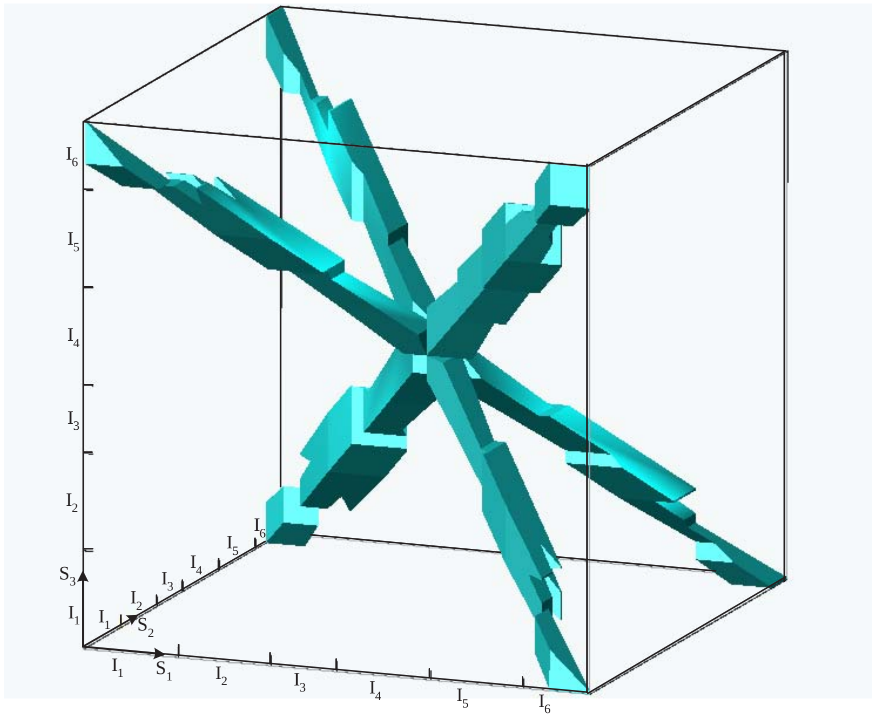


Figure 8: The collection of feasible 3-limb postures in contact c-space.

left and right sides of the tunnel are physically distinct. The eight outer vertices of contact c-space represent 3-limb postures where the three footpads are located at the bottom part of the tunnel. These vertices represent the $2^3 = 8$ possible assignments of the three limbs to the tunnel's two sides. Note that the robot must contact both sides of the tunnel in order to generate an equilibrium posture. Hence the vertices $(0,0,0)$ and $(540,540,540)$ certainly lie outside the set of feasible 3-limb postures presented below. Topologically, when one introduces a small cube-shaped puncture at the center point, contact c-space becomes a set embedded in a three-dimensional torus. This fact has been noted in the context of 3-finger grasps by Leveroni and Salisbury [13].

Let us now turn to the computation of the feasible 3-limb postures in contact c-space. Figure 7 shows the collection of 2-limb equilibrium postures in the (s_i, s_j) plane. It can be seen that these postures form a *convex polygon* in each planar cell. The edges of these polygons consist of frictional equilibrium constraints and the cell's boundaries. Note that the figure is symmetric with respect to the $s_i = s_j$ axis, reflecting the possibility of switching the limbs between the two contacts. Figure 8 shows the collection of feasible 3-limb postures. These postures are intersection of pairs of prisms whose polygonal cross section appears in Figure 7. In this particular tunnel, all prism intersections automatically satisfy the reachability constraint. (This is an artifact of our tunnel environment, coefficient of friction, and robot radius.) The collection of feasible 3-limb postures has a *six-fold symmetry* consisting of six symmetric "arms": every non-empty cell represents an assignment of the three limbs to a triplet of walls, and there are six such assignments. The arms are roughly aligned with the diagonals of contact c-space, and this can be explained as follows. The coordinate projection of each arm covers the entire length of the tunnel. Each arm can therefore be visualized as "dragging" the 3-limb mechanism as a single rigid body along the entire length of the

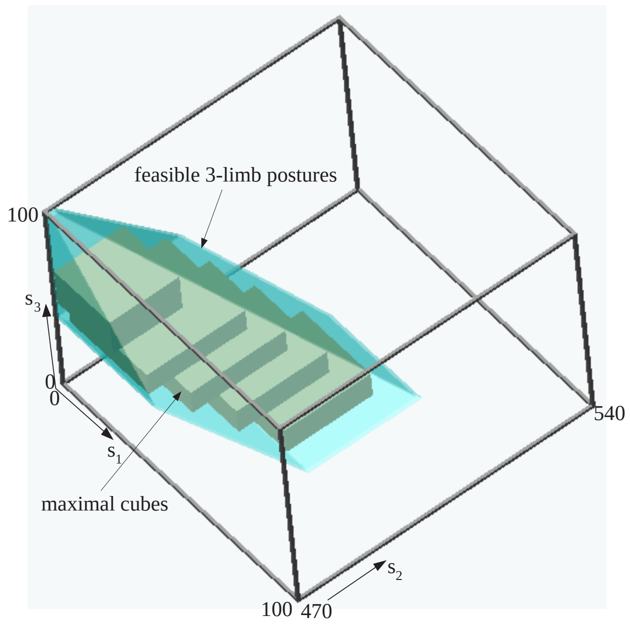


Figure 9: A five-cube approximation of the feasible 3-limb postures in the cell $I_1 \times I_6 \times I_1$.

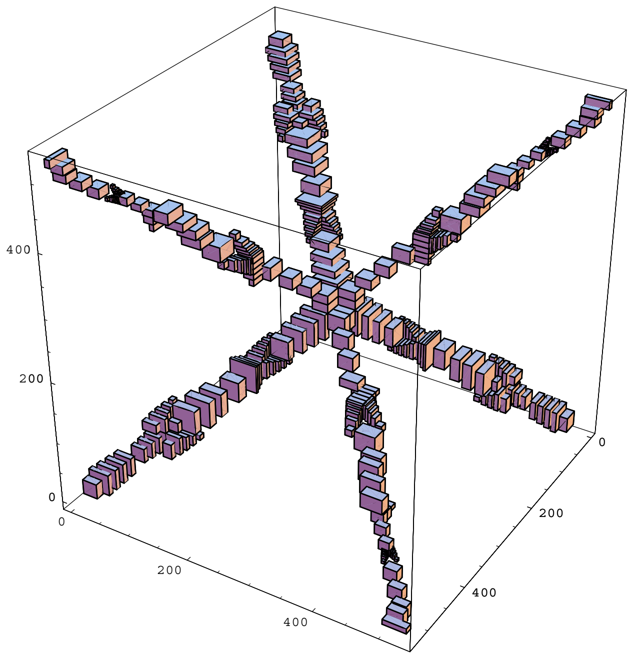


Figure 10: The collection of 270 maximal cubes approximating the feasible 3-limb postures.

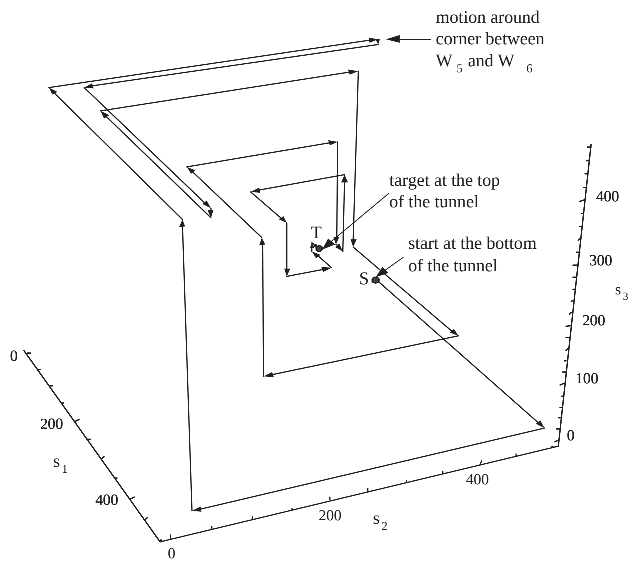


Figure 11: The shortest path from S to T along the sub-cube graph in contact c-space.

tunnel. There are nine non-empty cells in each arm, giving a total of 54 non-empty cells in the entire contact c-space.

Next consider the approximation of the feasible 3-limb postures in each cell by p maximal cubes. We use five maximal cubes per cell and compute the maximal cubes using the ellipsoid algorithm with an approximation error $\epsilon = 0.1$ (i.e. the volume of the maximal cubes is up to $1 + \epsilon$ times smaller than the volume of the exact maximal cubes). This value of $p = 5$ preserves the connectivity of the set of feasible 3-limb postures, while still being sufficiently low to allow reasonable execution time. Figure 9 shows the cube approximation of the feasible 3-limb postures in the cell $I_1 \times I_6 \times I_1$, where the relative configuration is specified by four separating planes orthogonal to the s_3-axis. The result of running the ellipsoid algorithm on the non-empty cells in one arm of contact c-space appear in Figure 10. Since there are 54 non-empty cells, the resulting cube approximation of contact c-space contains $5 \cdot 54 = 270$ maximal cubes. The algorithm next partitions each maximal cube along the separating planes of the other maximal cubes. The partitioning of the maximal cubes generated $230,900$ sub-cubes in contact c-space (the resulting sub-cubes are not shown).

The algorithm next constructs the sub-cube graph, assigns unit edge weights, and searches the graph for the shortest path from the start to target postures. The result of computing the shortest path using Dijkstra's algorithm is shown in Figure 11. Each segment in the figure is an edge of the sub-cube graph representing one limb lift-and-reposition step. Figure 12 shows the same path in physical space, where each foothold is marked by its index in the sequence of steps taken by the robot. Let us denote the sequence of 3-limb postures by (i_1, i_2, i_3), where i_j is the foothold position of limb j at the i^{th} stage. Then the path computed by the algorithm consists of the 3-limb postures: $S = (1,2,3) \to (4,2,3) \to (4,5,3) \to (4,5,6) \to (7,5,6) \to (7,8,6) \to (7,8,9) \to (7,10,9) \to (11,10,9) \to (11,10,12) \to (13,10,12) \to (13,14,12) \to (13,14,15) \to (16,14,15) \to (16,17,15) \to (16,17,18) \to (19,17,18) \to (19,20,18) \to (19,20,21) \to (22,20,21) \to (22,20,23) \to (22,24,23) \to (25,24,23) \to (25,24,26) \to (25,27,26) \to (28,27,26) \to (28,27,29) \to T = (30,27,29)$. This sequence describes a 3-2-3 gait pattern, where successive 3-limb postures are interspersed by a 2-limb posture that allows motion of a limb between the two 3-limb postures.

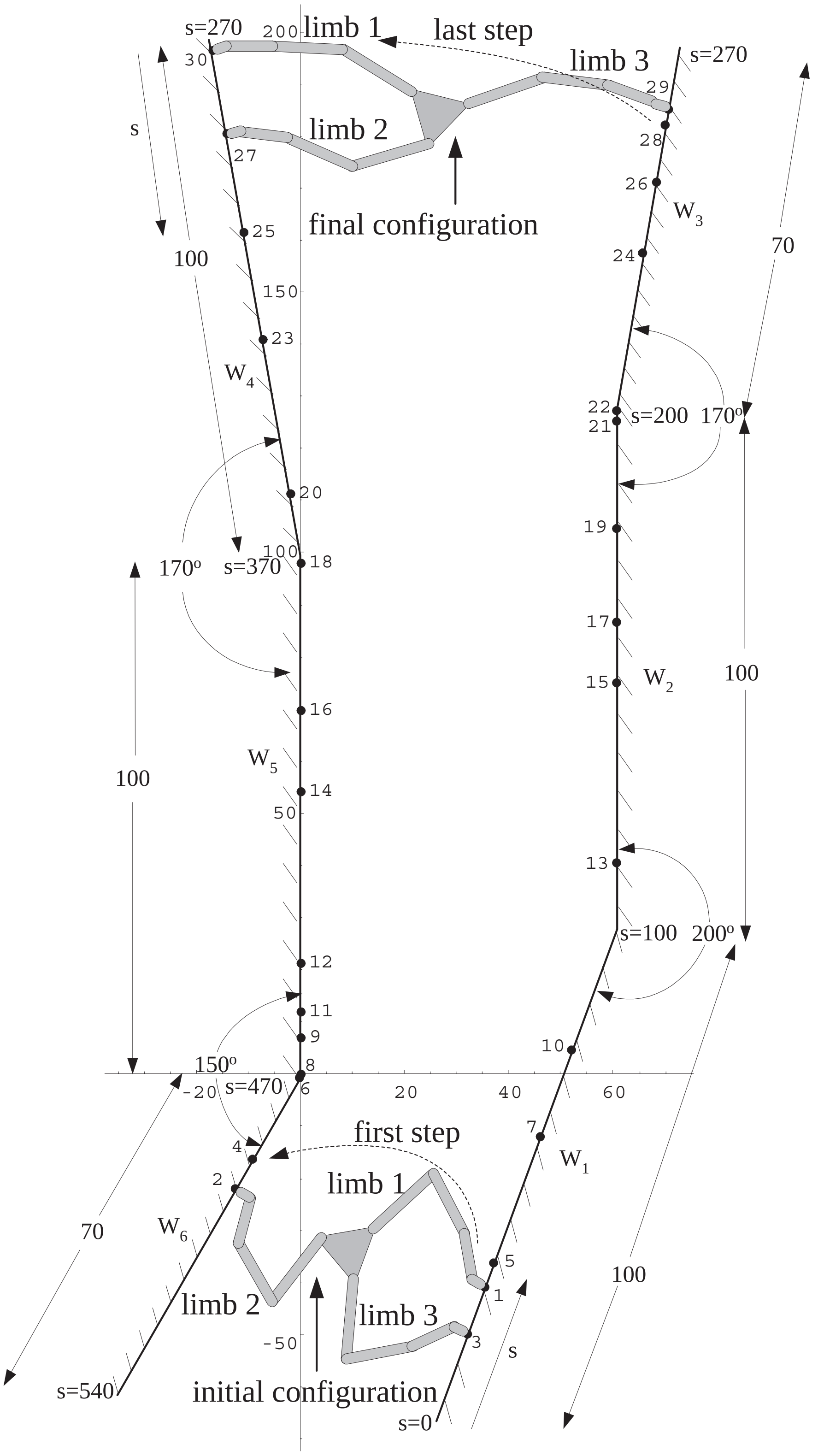


Figure 12: The tunnel environment used in the simulations, and the sequence of footholds generated by the PCG algorithm.

Note that the path generated by the algorithm is minimal in terms of the number of steps relative to the cube approximation of the feasible 3-limb postures (Figure 10). Note, too, that the short edge along the s_3 axis in Figure 11 corresponds to the transition $(7, 8, 6) \to (7, 8, 9)$. This edge takes the robot around the corner between the walls W_5 and W_6 in Figure 12. The difficulty in accomplishing this maneuver can be appreciated by inspecting the narrow overlap between the planar cells $I_5 \times I_1$ and $I_6 \times I_1$ in Figure 7.

Some remarks on the algorithm run time: We ran the algorithm on a a Pentium 1.7GHz computer with 512MB of RAM. The algorithm was programmed in Mathematica[4] native code and it was not compiled into C. Due to the size of the sub-cube graph which contains $230,900$ nodes, searching the took most of the runtime. In the search for shortest path we used a naive Dijkstra's algorithm. The search took $117,500$ iterations. Additional limitation was the computer amount of memory. While running the algorithm the computer also ran the operating system and other resident programs in the background. These programs used at least half the memory of the computer. Thus due to the size of the database the operating system was forced to use the Hard Disk for on-line memory usage. Under these conditions running the example presented here took about 78 hours. We suspect that more efficient search algorithm that will be compiled in C can dramatically reduce the algorithm's runtime.

6 Experimental Results

This section describes an experimental implementation of the PCG algorithm. The objective of the experiments is to test the robustness of the PCG algorithm with respect to small footpad placement errors. We used for this purpose a 3-limbed robot having four revolute joints per limb, giving a total of twelve actuated degrees of freedom (Figure 1). The limbs are attached to a central base which has three unactuated degrees of freedom. The distance from the robot's center to the tip of a fully stretched limb is $R = 90$ cm. The robot operates in a horizontal plane, and is supported against gravity by air bearings mounted under the central base and the distal link of each limb. The robot's twelve joints are instrumented with 2000 count-per-revolution optical encoders (mounted on the joint axis rather than the motor shaft), and the robot's motion is controlled with MEI controllers that allow decentralized PID control of the twelve axes. However, the robot has no visual feedback from the environment, and is not equipped with any contact force sensors. The robot was placed on a horizontal plane bounded by piecewise linear walls made of stiff Plexiglas (Figure 13). The walls were coated with medium rough sand paper, giving a friction coefficient of at least $\mu = 0.5$. Note that the tunnel layout provides various geometric features such as parallel, diverging, and converging wall segments.

The PCG algorithm was given a geometric description of the tunnel, as well as the start and target 3-limb postures depicted in Figure 13. Based on this information, the algorithm computed the sequence of footholds marked 1 to 10 along the tunnel walls in Figure 13. The corresponding minimum cost path along the sub-cube graph is shown in Figure 14. At the next stage the sequence of footholds was converted to continuous central-base and joint trajectories that were fed to the robot's controllers. (The details of this stage are not considered in this paper, but the central-base and joint trajectories were computed as to satisfy reachability and avoid collision with the tunnel walls and between limbs.) A major

[4]Mathematica is a trade mark of Wolfram Research, Inc.

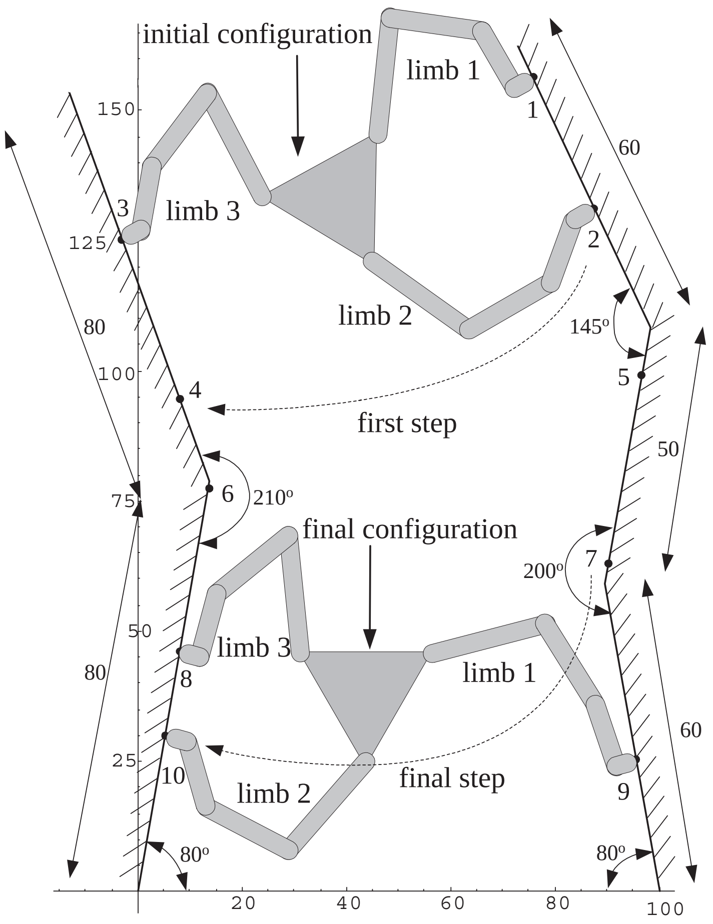


Figure 13: The tunnel environment used for the experiment. The footholds computed by the algorithm are numbered 1 to 10.

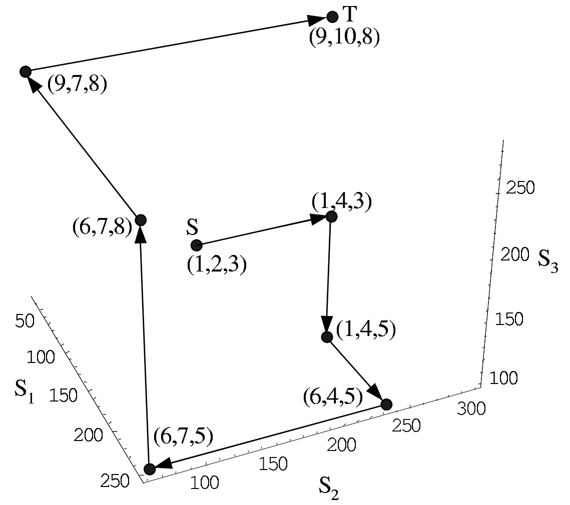


Figure 14: The minimum cost path from S to T along the edges of the sub-cube graph. Each node is marked with the walls being contacted by (limb 1, limb 2, limb 3).

source of position errors in our implementation has been the coarse granulation of the joint trajectories, which were discretized and fed to the motor controllers at a rate of 10 set points per second.

It is worth to mention that in this example the algorithm generated footholds positions which are in the proximity of tunnel corners (e.g points 6,8,18,21,22). If necessary it is possible to avoid such footholds by virtually shortening the tunnel segments. Thus, each tunnel segment will end up in a safe distance from the real corner. Note that the PCG algorithm does not require the tunnel to be connected.

The experiment started with a calibration process during which the robot positioned itself at the start posture. Then motion started, and the time history of the twelve axes as well as the central-base three configuration parameters were recorded. Figure 15 shows the robot's configuration history during the entire motion from S to T. The first row shows the x, y, θ coordinates of the central base. The remaining rows show the joint angles of each limb, where the leftmost graph describes the proximal joint and the rightmost graph describes the distal joint. The total length of the robot's central-base trajectory along the y coordinate was 1.5 m, and the total motion time was 33 minutes. This low speed motion allowed a fairly accurate tracking of the pre-designed path in $I\!R^{15}$, without using any feedback on the central-base position and orientation. The tracking accuracy can be seen in the graphs as very small deviations between the desired and actual paths. Note that a considerably larger deviation occurred during the last few minutes in the distal joint of the second limb. This limb established contact with the tunnel wall before its distal joint reached the desired angle. While applying contact force against the tunnel wall, the distal motor saturated on its maximal output torque without being able to reach its desired angle. This error was eventually corrected when the limb broke contact with the wall and moved to the next foothold position. Snapshots from a video of the robot motion during the experiment appear in Figure 16.

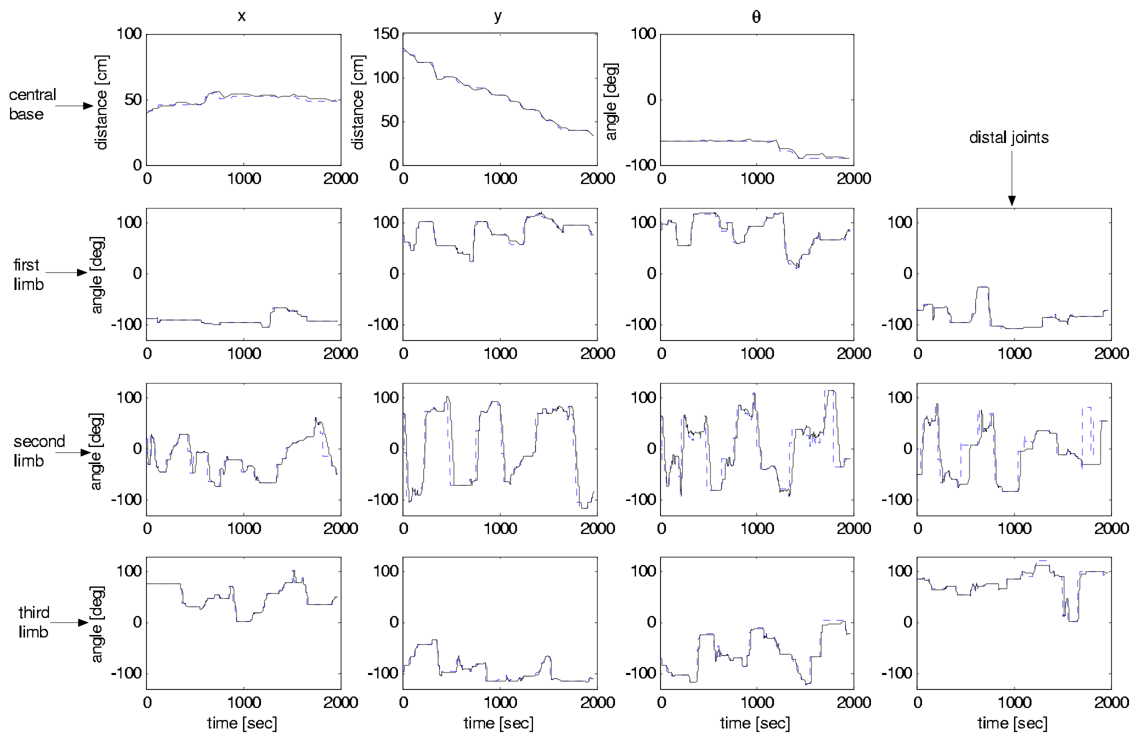


Figure 15: Measurements of the robot's 15 configuration parameters during the experiment. The nominal paths are indicated as dashed lines, and the actual measurements are indicated as solid lines.

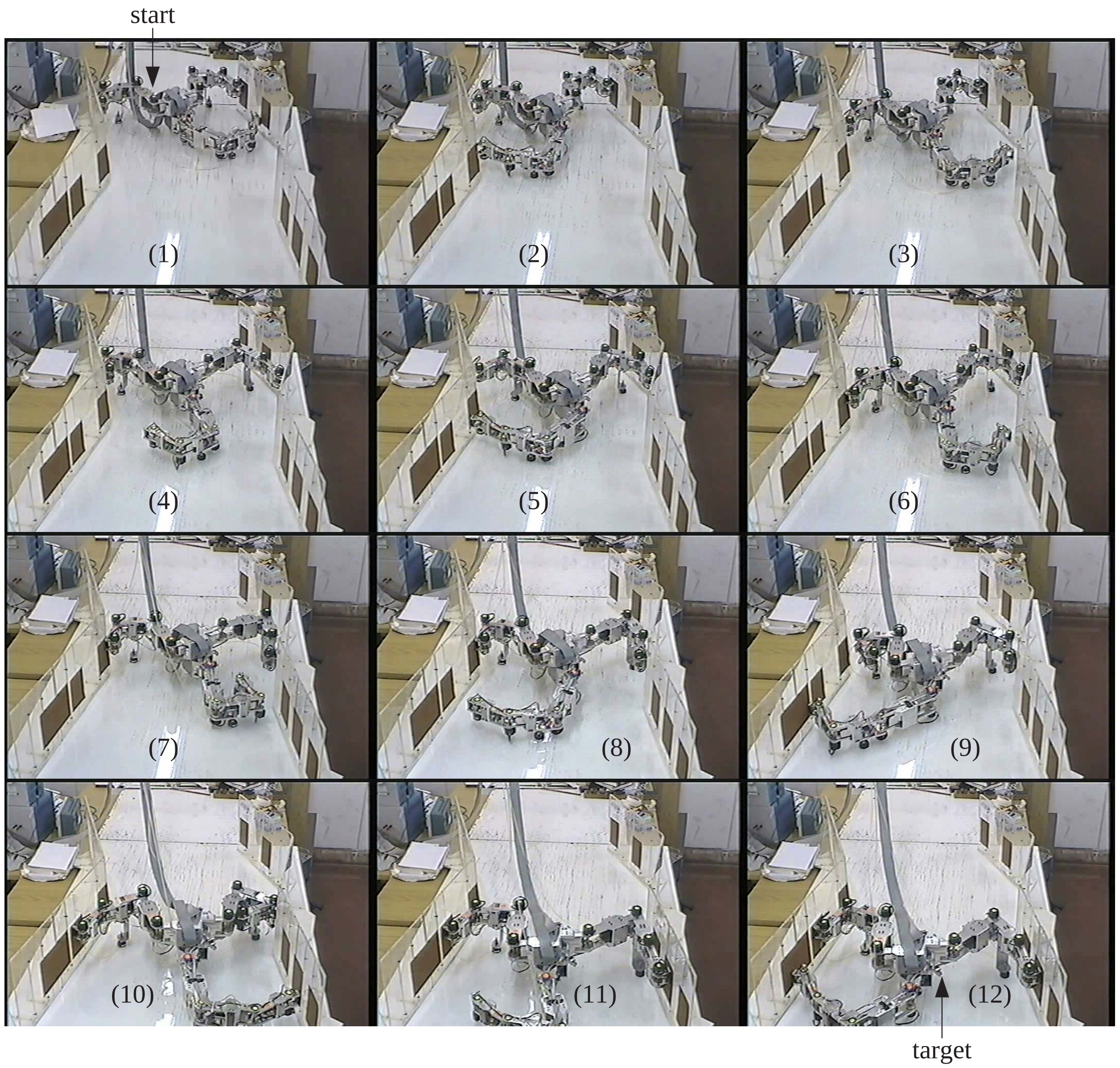


Figure 16: Snapshots from the video showing the spider robot moving in the tunnel (see http://robots.technion.ac.il for the full video).

7 Conclusion

We presented an algorithm for selecting the foothold positions of a 3-limb robot in planar tunnel environments. The algorithm assumes knowledge of the tunnel geometry and a lower bound on the amount of friction at the contacts. Using this knowledge, we established that the feasible 3-limb postures consist of a union of convex sets in contact c-space. Using convex programming techniques, the algorithm approximates the feasible 3-limb postures by $O(np)$ maximal cubes, where n is the number of tunnel walls and p is the number of maximal cubes which approximate each convex set in contact c-space. The algorithm next partitions the cubes into sub-cubes, and defines a graph whose nodes are sub-cubes and whose edges represent feasible motion of a limb between any 3-limb postures in the two sub-cubes. A search for the minimum cost path in the graph generates a 3-2-3 gait sequence that moves the robot from start to target while minimizing the number of foothold exchanges along the path. The algorithm has been demonstrated in a simulated tunnel as well as in actual experiments with a 3-limb robot prototype.

The algorithm's main strength is its emphasize on computing contact independent foothold placement sequences. Each sub-cube parameterizes three contact independent wall segments, and motion along an edge of the sub-cube graph can be realized by limb reposition between any two postures in the two sub-cubes connected by the edge. Thus a controller for the robot's limbs need only ensure footpad placement within the segments parameterized by the sub-cubes. An example for such controller is the decentralized PD position controller that was used in our experiments. The algorithm's main weakness is the lack of a systematic procedure for selecting the input parameter p. We described a preliminary approach for selecting p based on the requirement that the cube approximation preserve the connectivity of the feasible 3-limb postures.

Finally let us mention two possible generalizations of the algorithm. The first generalization is to 3-limb spider robots that move under the influence of gravity in two-dimensions. In this case the position of the robot's center of mass plays an important role in determining posture stability, and contact c-space must be augmented with two parameters representing the center-of-mass positions. Research under progress indicates that the feasible equilibrium postures under gravity still form convex sets. The algorithm is therefore extendible to motion under the influence of gravity. The second generalization is to planar robots having a larger number of limbs. It seems that the algorithm directly generalizes to k-limb mechanisms that move in tunnel environments using a k–$(k-1)$–k gait pattern. Contact c-space in this case is k-dimensional, and the feasible k-limb postures seem to be a union of convex sets. In that case the algorithm can be applied to k-limb mechanisms without any change. The computational complexity of the k-limb algorithm would become $O(np^{k+3}\log(np))$ i.e., exponential in the number of limbs. A more challenging topic is how to plan the foothold positions of a k-limb mechanism using a variable gait pattern. For instance, a 4-limb mechanism can move a single limb at a time, or two limbs at a time, resulting in a variable gait pattern. We are currently investigating foothold placement algorithms for such mechanisms, with the objective of generating polynomial time algorithms that minimize the number of steps from start to target while exploiting variable gait patterns.

A Details of the PCG Algorithm

This appendix describes several details of the PCG algorithm. First we describe the necessary modification to the algorithm when a cell contains two or three possibly overlapping convex sets of feasible 3-limb postures. Since each convex set is approximated individually by p maximal cubes, it is possible that two maximal cubes originating from different convex sets would overlap in contact c-space. However, each cube still has its own unique orientation vector. The partitioning of the maximal cubes into sub-cubes proceeds as before. The edges between sub-cubes are assigned a weight of unity or zero according to the following two cases. If two sub-cubes connected by an edge are disjoint, the edge is assigned a *unit* weight as before. In the second case the edge connects two copies of the same sub-cube. We represent the two sub-cubes as distinct nodes, and assign *zero* weight to the edge connecting the two sub-cubes. Note that zero-weight edges provide important pathways in the sub-cube graph. Rather than representing a physical limb lifting and reposition, these edges represent a freedom of the algorithm to select among more than one limb for its next step.

The following lemma asserts that motion along an edge of the sub-cube graph can be realized by a continuously reachable limb motion.

Lemma A.1. *Consider two reachable 3-limb postures. If two limbs and their footpad positions are common to both postures, there exists a path that takes the third limb between the two postures such that the three footpads are continuously reachable along the path.*

The lemma generalizes as follows. If two k-limb postures share at least two limbs and their contacts, there exists a path for the remaining $k-2$ limbs between the two postures such that all k footpads are continuously reachable along the path.

Proof: The minimum-radius discs containing the two triplets of foothold positions necessarily overlap, since two foothold positions are common to both postures. The radius of the two discs is bounded by R, since the two triplets of foothold positions are reachable. It follows that any motion of the third limb between its two footholds, such that its footpad lies in the union of the two discs, guarantees that the three footpads, one moving and two stationary, are continuously reachable along the path. □

The next lemma gives necessary and sufficient conditions for target reachability.

Lemma 4.1. *Let S and T be start and target 3-limb postures. A sufficient condition for $T \in \mathcal{R}(S)$ is that the sub-cube graph contains a path from S to T whose nodes change orientation at least once along the path.*
The sufficient condition is also necessary when T lies outside the planes passing through S and orthogonal to the axes of contact c-space.

Proof: We consider only the sufficient condition. Let $\mathcal{D}_1, \ldots, \mathcal{D}_p$ be the sub-cubes corresponding to the nodes of the sub-cube graph along the path from S to T. For simplicity we assume that $S \in \mathcal{D}_1$ and $T \in \mathcal{D}_p$. Let u denote the orientation vector of $\mathcal{D}_1$, and let P denote the plane orthogonal to u and passing through S. Let $\mathcal{D}_{i+1}$ be the sub-cube experiencing the first orientation change, so that $\mathcal{D}_1, \ldots, \mathcal{D}_i$ have an orientation vector parallel to u. Then $\mathcal{D}_i \cap \mathcal{R}(S)$ is the rectangle $\mathcal{D}_i \cap P$. Since the orientation vectors are aligned with the axes of contact c-space, $\mathcal{D}_{i+1}$ has an orientation vector, denoted v, which is orthogonal to u and therefore lies in P. Since the sub-cube graph contains an edge between $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$, the two sub-cubes are compatible. It follows that the subset of $\mathcal{D}_{i+1}$ reachable from $\mathcal{D}_i \cap \mathcal{R}(S)$ along a single-step motion is the rectangle $\mathcal{D}_{i+1} \cap P$. Since v lies in P, any motion starting in $\mathcal{D}_{i+1} \cap P$ in the direction orthogonal to this rectangle is admissible. The totality of these motions fills the sub-cube $\mathcal{D}_{i+1}$. From this stage onward $\mathcal{R}(S)$ fills the remaining sub-cubes $\mathcal{D}_{i+2}, \ldots, \mathcal{D}_p$. Since $T \in \mathcal{D}_p$, the target belongs to $\mathcal{R}(S)$. □

The following proposition characterizes the algorithm's optimality.

Proposition 4.2. *Let $\mathcal{X}$ be an approximation of the feasible 3-limb postures by maximal cubes. Let S and T be start and target 3-limb postures such that $T \in \mathcal{R}(S)$. Then up to one extra step, the path computed by the PCG algorithm minimizes the number of steps over all paths from S to T in $\mathcal{X}$.*

Proof: We consider the case where S and T both lie in $\mathcal{X}$. Recall that paths in contact c-space are rectilinear with each segment representing one limb reposition step. Let α be a path from S to T in $\mathcal{X}$ having the minimum number of segments, and let $N(\alpha)$ be the number of its segments. Let β be a minimum cost path from S to T computed by the algorithm along the sub-cube graph. We wish to show that β can be realized by a rectilinear path γ in contact c-space whose number of segments, $N(\gamma)$, satisfies the inequality $N(\gamma) \leq N(\alpha) + 1$. As an intermediate step, we show that β can be realized by a path γ such that $N(\gamma)$ equals to the number of nodes along β. The proof is by induction on the number of nodes along β. The induction starts with paths having three nodes, but first we remark on paths having fewer nodes. When β has a single node, S and T lie in the sub-cube corresponding to this node. Since $T \in \mathcal{R}(S)$, the target can be reached along a path γ having at most *two* segments, which is the minimum number of steps. When β passes through two nodes, S and T lie in the two sub-cubes corresponding to these nodes. Since $T \in \mathcal{R}(S)$, the target can be reached along a path γ having at most *three* segments, which is again the minimum number of steps.

Consider now the case where β passes through three nodes. We wish to show that β can be realized by a path γ having three segments. Let $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ denote the sub-cubes corresponding to the three nodes, and let v_1, v_2, v_3 denote their orientation vectors. Then $S \in \mathcal{D}_1$ and $T \in \mathcal{D}_3$. Let P be the plane passing through S and orthogonal to v_1, and let P' be the plane passing through T and orthogonal to v_3. There are two sub-cases to consider. In the first sub-case v_1 is parallel to v_3 or, equivalently, P is parallel to P'. Let l_1 be any segment between $\mathcal{D}_1$ and $\mathcal{D}_2$, and let l_2 be any segment between $\mathcal{D}_2$ and $\mathcal{D}_3$. Since β is a minimum cost path, l_1 and l_2 cannot be collinear, as this would imply that $\mathcal{D}_1$ is directly connected to $\mathcal{D}_3$ in the sub-cube graph. By construction l_1 is orthogonal to v_1 and v_2, while l_2 is orthogonal to v_2 and v_3. Since v_1 and v_3 are parallel, they are orthogonal to the plane spanned by l_1 and l_2. Thus v_1, v_2, and v_3 are all parallel. Since $T \in \mathcal{R}(S)$, the plane P' must coincide with P. Hence T can be reached along a path γ having at most *two* segments embedded in this plane. Consider now the sub-case where v_1 and v_3 are non-parallel. Let l_1 be the particular segment starting at $S \in \mathcal{D}_1$ and leading into $\mathcal{D}_2$, and let l_2 be the particular segment ending at $T \in \mathcal{D}_3$ and originating from $\mathcal{D}_2$. Then either l_1 is collinear with v_3, or l_2 is collinear with v_1. If l_1 is collinear with v_3, it must hit the rectangle $\mathcal{D}_2 \cap P'$. If l_2 is collinear with v_1, it must hit the rectangle $\mathcal{D}_2 \cap P$. In either case γ can be realized using *three* segments. For instance, when l_1 is collinear with v_3, the first segment is l_1, the second and third segments are embedded in P' and connect the endpoint of l_1 in $\mathcal{D}_2 \cap P'$ to T.

Next consider the induction step, where β passes through $i > 3$ nodes corresponding to sub-cubes $\mathcal{D}_1, \ldots, \mathcal{D}_i$ such that $S \in \mathcal{D}_1$ and $T \in \mathcal{D}_i$. There are two cases to consider. In the first case $\mathcal{R}(S)$ fills the entire sub-cube $\mathcal{D}_{i-1}$. The induction hypothesis implies that any point in $\mathcal{D}_{i-1}$ can be reached from S along at most $i-1$ steps. Since the sub-cube graph contains an edge between $\mathcal{D}_{i-1}$ and $\mathcal{D}_i$, this edge can be realized by a single segment from $\mathcal{D}_{i-1}$ to T in $\mathcal{D}_i$. Hence T can be reached along a path γ having at most i segments. In the second case $\mathcal{R}(S)$ fills only a subset of $\mathcal{D}_{i-1}$. Lemma 4.1 implies that in this case the orientation vectors of $\mathcal{D}_1, \ldots, \mathcal{D}_{i-1}$ are all parallel to v_1. The reachable subset of $\mathcal{D}_{i-1}$ is therefore the rectangle $\mathcal{D}_{i-1} \cap P$, where P is the plane defined above. Since the portion of γ until $\mathcal{D}_{i-1}$ is embedded in P, this portion of γ can be realized by a path having one segment per transition between successive sub-cubes, which gives $i-2$ segments from $S \in \mathcal{D}_1$ to $\mathcal{D}_{i-1}$. There are now two sub-cases to consider, according to the position of T relative to P. We consider only the sub-case where T lies outside P. Since $T \in \mathcal{R}(S)$, the orientation vector of $\mathcal{D}_i$ must be rotated with respect to v_1. Letting P' be the plane passing through T and orthogonal to the orientation vector of $\mathcal{D}_i$, the line $l = P \cap P'$ contains a valid segment from $\mathcal{D}_{i-1}$ to $\mathcal{D}_i$. The latter segment reaches the rectangle $\mathcal{D}_i \cap P'$, from which a segment orthogonal to l reaches T. Thus we added two segments to the path from S to $\mathcal{D}_{i-1}$, generating a path γ containing at most i segments. To summarize, β can be realized by a rectilinear path γ whose number of segments $N(\gamma)$ equals the number of nodes along β.

The sub-cubes form a partition of $\mathcal{X}$. Since the minimum cost path α is rectilinear with vertices in $\mathcal{X}$, every vertex of α lies in some sub-cube. If two successive vertices of α lie in distinct sub-cubes, the segment joining the two vertices is parallel to some s_i-axis and its direction necessarily respects the orientation vector of both sub-cubes. It follows that the sub-cube graph contains an edge between the nodes corresponding to the two sub-cubes. By construction β minimizes the number of edges from S to T along the sub-cube graph. Hence the number of edges along β, denoted N, satisfies $N \leq N(\alpha)$. The number of nodes along β is $N+1$. Since β can be realized by a path γ having $N(\gamma) \leq N+1$ segments, we obtain that $N(\gamma) \leq N(\alpha)+1$. $\square$

B Selection of Number of Maximal Cubes

In this appendix we suggest a method for selecting the number of maximal cubes, p, which approximate each convex set of feasible 3-limb postures. The method is based on the following two-stage approach. First we determine the connectivity relation between the convex sets in contact c-space. Then we describe a decision algorithm that determines if p preserves the above connectivity relation as well as the connectivity within each convex set. We begin with a characterization of the required connectivity check. By definition, two convex sets of feasible 3-limb postures are *connected* if they have an overlapping projection along a direction perpendicular to the orientation vector of both sets. The following lemma asserts that this type of connectivity can be checked as a convex programming problem.

Lemma B.1. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two convex sets in $\mathbb{R}^n$, and let $\mathcal{I}$ be a set of coordinate indices of $\mathbb{R}^n$. Then the existence of points $x = (x_1, \ldots, x_n) \in \mathcal{S}_1$ and $y = (y_1, \ldots, y_n) \in \mathcal{S}_2$ such that $x_j = y_j$ for $j \in \mathcal{I}$ is a convex programming problem.*

Proof: First we embed $\mathcal{S}_1$ and $\mathcal{S}_2$ in the orthogonal subspaces $\mathbb{R}^n \times 0$ and $0 \times \mathbb{R}^n$ of $\mathbb{R}^{2n}$. Next we construct $2n$-dimensional cylinders over the embedded sets, denoted S_1 and

S_2. These cylinders are convex in $\mathbb{R}^{2n}$, since $\mathcal{S}_1$ and $\mathcal{S}_2$ are convex in $\mathbb{R}^n$. Every point in $S_1 \cap S_2$ represents a pair of points $x, y \in \mathbb{R}^n$ such that $x \in \mathcal{S}_1$ and $y \in \mathcal{S}_2$. Consider now the linear subspace in $\mathbb{R}^{2n}$ given by $P = \{(s_1, \ldots, s_{2n}) : \; s_j = s_{j+n} \, \forall j \in \mathcal{I}\}$. Every point in P represents two points in $\mathbb{R}^n$ that share the same j^{th} coordinate for all $j \in \mathcal{I}$. The set $P \cap S_1 \cap S_2$ is convex, since it is an intersection of convex sets. Moreover, every point in $P \cap S_1 \cap S_2$ represents a pair of points $x \in \mathcal{S}_1$ and $y \in \mathcal{S}_2$ that share the j^{th} coordinate for all $j \in \mathcal{I}$. The problem has now become a standard check that a given convex set is non-empty, which is a convex optimization problem (e.g. [4, p. 329]). $\square$

Next we describe a decision algorithm which verifies that p is sufficiently large, based on the criterion that the cube approximation should preserve the connectivity of the feasible 3-limb postures in contact c-space. Once a sufficiently large p is found, a binary search based on the decision algorithm can give the smallest p that preserves connectivity.

```
An algorithm for checking that $p$ preserves connectivity
Input: A cube approximation of the feasible 3-limb postures based on $p$ maximal cubes per
          non-empty convex set.
For each convex set $\mathcal{F}_{ijk}$ with orientation vector along $s_l$-axis do:
   1. Check if maximal cubes in $\mathcal{F}_{ijk}$ have a common projection along axis perpendicular
    to $s_l$. If not, STOP, $p$ is too small.
   2. Find all $\mathcal{F}_{mno}$ such that
    Case $l = 1$: $m = i$ and $(n = j$ or $o = k)$
    Case $l = 2$: $n = j$ and $(m = i$ or $o = k)$
    Case $l = 3$: $o = k$ and $(m = i$ or $n = j)$
   3. For every $\mathcal{F}_{mno}$ from step 2 do:
    3.1 Check if $\mathcal{F}_{ijk}$ overlaps $\mathcal{F}_{mno}$ using convex programming. If there is no overlap,
    continue to next $\mathcal{F}_{mno}$ in step 3.
    3.2 Check compatability of all possible pairs of maximal cubes such that one cube is
    in $\mathcal{F}_{ijk}$ and the other is in $\mathcal{F}_{mno}$. If there is no compatible pair, STOP, $p$ is too small.
End of do loop.
```

The effect of p on the cube approximation is illustrated in Figure 17. The figure depicts a two-dimensional cube approximation analogous to our three-dimensional case. The figure shows that connectivity is preserved by the maximal cubes only for $p \geq 3$. The figure suggests that there is no significant benefit in increasing p beyond a certain value, since improvement in path cost is only minor. Finally, we ran simulations to investigate the effect of p on the number of steps along the path from S to T. We used a 3-limb robot with reachability radius of $R = 77$ cm, and tested two tunnels having a friction coefficient of $\mu = 0.5$. The first tunnel consists of two parallel walls of length 200 cm and width 116 cm. The second tunnel consists of a straight wall of length 150 cm, and a wall composed of two segments 80 cm long at an angle of 150°. The sharp angle between the wall segments points into the tunnel, and the narrowest part of the tunnel is 102 cm wide. The results of running the PCG algorithm for $p = 2, \ldots, 6$ in the two tunnels are listed in the following table. The table shows that once p is sufficiently large to yield a path from S to T, subsequent increase of p gives only a minor reduction in the number of steps along the path.

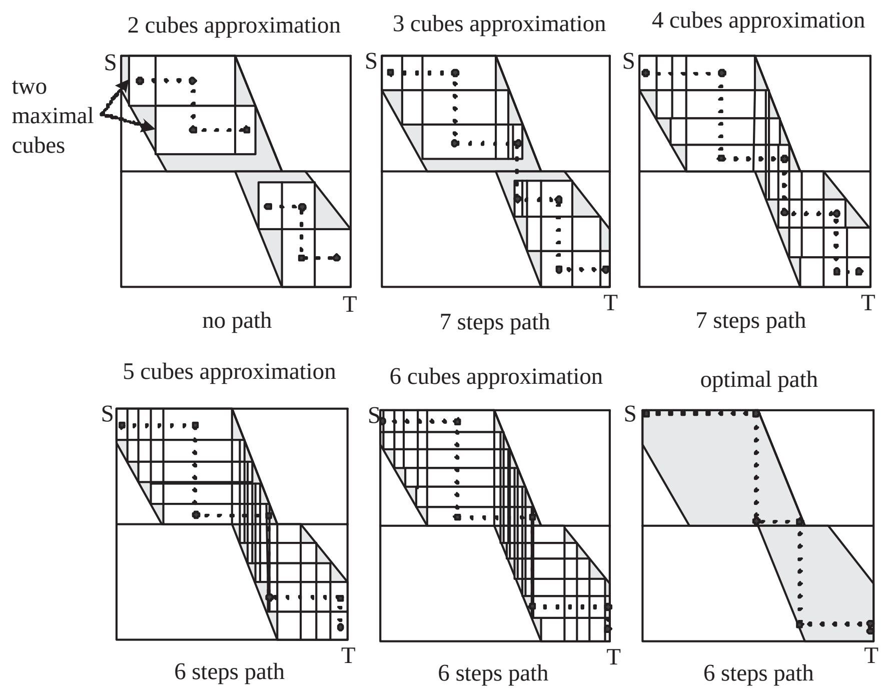


Figure 17: The effect of p on the number of steps along the path from S to T.

p	number of steps in | | tunnel	number of steps in > | tunnel
2	no path	no path
3	10	9
4	10	9
5	10	8
6	10	7

C Index to Multimedia Extensions

The multimedia extensions to this article are at: http://www.ijrr.org.

Extension	Type	Description
1	Video	Spider robot motion in a tunnel

The accompanied video present the prototype of our 3-limbed robot conducting the experiment discussed in Section 6 and walks through a $2D$ tunnel. This video shows the applicability of the algorithm presented in this paper to a real world problem. The algorithm gets as an input the tunnel geometry, the amount of friction between the robot footpads and the walls, and teachability radius of the robot. Next it off-line compute a sequence of foothold positions. A continuous central-base and joint trajectory is computed off-line, and fed to the robot's controller. Since the mechanical design of the robot prevent from one limb to cross over the other a set of limb replacement maneuvers is needed. These maneuvers can be seen in the video as one limb contacts the wall, the another limb moves towards the same contact point and finally contact the wall near the first limb. At this stage where two limbs contact the wall in two very close points the first limb disjoint the wall and move forward to conduct the next step. These maneuvers enlarge the total number of steps conducted by the mechanism, but they are caused only due to mechanical limitations. Finally, it is possible to design a mechanism that overcome this mechanical limitation and that can fully perform the steps produced by the algorithm.

References

[1] A. Bicchi. Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE Trans. on Robotics and Automation*, 16(6):652–662, 2000.

[2] J.-D. Boissonnat, O. Devillers, L. Donati, and F. P. Preparata. Motion planning for spider robots. In *IEEE Int. Conf. on Robotics and Automation*, pages 2321–2326, 1992.

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

[4] S. P. Boyd and C. H. Barratt. *Linear Controller Design.* Prentice Hall, NJ, 1991.

[5] T. Bretl, S. Rock, and J. C. Latombe. Motion planning for a three-limbed climbing robot in vertical natural terrein. In *IEEE Int. Conf. on Robotics and Automation*, pages 2946–2953, Taipei, Taiwan, September 2003.

[6] F. H. Clarke. *Optimization and Nonsmooth Analysis.* SIAM Publication, 1990.

[7] B. Goodwine and J. W. Burdick. Motion planning for kinematic stratified systems with application to quasi–static legged locomotion and finger gaiting. *IEEE Transactions on Robotics and Automation*, 18(2):209–222, 2002.

[8] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *IEEE Int. Conf. on Robotics and Automation*, pages 1321–1326, 1998.

[9] S. Hirose and O. Kunieda. Generalized standard foot trajectory for a quadruped walking vehicle. *The Int. J. of Robotics Research*, 10(1):2–13, 1991.

[10] J. Hong, G. Lafferriere, B. Mishra, and X. Tan. Fine manipulation with multifinger hands. In *IEEE Int. Conf. on Robotics and Automation*, pages 1568–1573, 1990.

[11] D. E. Koditschek. An approach to autonomous robot assembly. *Robotics*, 12(2):137–155, 1994.

[12] J. K. Lee and S. M. Song. Path planning and gait of walking machine in an obstacle-strewn environment. *J. Robotics Systems*, 8:801–827, 1991.

[13] S. Leveroni and K. Salisbury. Reorienting objects with a robot hand using grasp gaits. In *7th Int. Symp. on Robotics Research*, pages 2–15, 1995.

[14] A. Madhani and S. Dubowsky. Motion planning of mobile multi-limb robotic systems subject to force and friction constraints. In *IEEE Int. Conf. on Robotics and Automation*, pages 233–239, 1992.

[15] D. W. Marhefka and D. E. Orin. Gait planning for energy efficiency in walking machines. In *IEEE Int. Conf. on Robotics and Automation*, pages 474–480, 1997.

[16] Y. E. Nesterov and A. S. Nemirovsky. *Interior Point Polynomial Methods in Convex Programming: Theory and Applications.* Springer Verlag, New York, 1992.

[17] V.-D. Nguyen. Constructing force-closure grasps. *The Int. Journal of Robotics Research*, 7(3):3–16, 1988.

[18] D. O'Halloran, A. Wolf, and H. Choset. Design of a high-impact survivable robot. In *IEEE Int. Conf. on Robotics and Automation*, pages 3551–3558, 2004.

[19] F. Pfeiffer, T. Rossmann, N. Bolotnik, F. Chernousko, and G. Kostin. Simulation and optimization of regular motions of a tube-crawling robot. *Multibody System Dynamics*, 5:159–184, 2001.

[20] J. Ponce and B. Faverjon. On computing three-finger force closure grasps of polygonal objects. *IEEE Trans. on Robotics and Automation*, 11(6):868–881, 1995.

[21] J. Ponce, D. Stam, and B. Faverjon. On computing force-closure grasps of curved two-dimensional objects. *The International Journal of Robotics Research*, 12(3):263–273, June 1993.

[22] J. Savall, A. Avello, and L. Briones. Two compact robots for remote inspection of hazardous areas in nuclear power plants. In *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pages 1993–1998, 1999.

[23] A. Shapiro. *Design and Control of An Autonomous Spider-Like Robot for Motion in 2D Tunnel Environments.* PhD thesis, ME Dept. Technion, http://robots.technion.ac.il, October 2003.

[24] T. J. Stone, D. S. Cook, and B. L. Luk. ROBUG iii - the design of an eight legged teleoperated walking and climbing robot for disordered hazardous environments. *Mechanical Incorporated Engineer*, 7(2):37–41, 1995.

[25] K. van der Doel and D. K. Pai. Performance measures for locomotion robots. *J. of Robotic Systems*, 14(2):135–147, 1997.

[26] Y. Wei and B. Goodwine. Stratified motion planning on non-smooth domains with application to robotic legged locomotion and manipulation. In *IEEE Int. Conf. on Robotics and Automation*, pages 3546–3551, 2002.

[27] T. Yoshikawa. Passive and active closure by constraining mechanisms. *J. of Dynamic Systems, Measurement, and Control*, 121 (3-4):418–424, 1999.

[28] A. Zagler and F. Pfeiffer. Moritz a pipe crawler for tube junctions. In *IEEE Int. Conf. on Robotics and Automation*, pages 2954–2959, Taipei, Taiwan, September 2003.