Range-Sensor-Based Navigation in Three-Dimensional Polyhedral Environments

Ishay Kamon

CS Department
Technion—Israel Institute of Technology
Technion City, Israel

Elon Rimon

ME Department
Technion—Israel Institute of Technology
Technion City, Israel

Ehud Rivlin

CS Department
Technion—Israel Institute of Technology
Technion City, Israel

Abstract

This paper is concerned with the problem of sensor-based navigation in three dimensions. The robot, which is modeled as a "bug" or a "point robot," has no a priori knowledge of the environment. It must rather use its sensors to perceive the environment and plan a collision-free path to various targets. The robot is further required to navigate in the most reactive way possible, retaining the smallest amount of information required for global convergence to the target. We assume a three-dimensional polyhedral environment and present two basic results for sensor-based navigation in this environment. First we establish sufficient conditions for range-sensor-based exploration of the entire surface of a general polyhedron and present a strategy for performing this task. Then we characterize the locally shortest path from the current robot location to the target and present a method for estimating this path in time that is linear with the problem size. Based on these results, we present a range-sensor-based navigation algorithm for a bug robot in a general three-dimensional polyhedral environment. The algorithm, called 3DBug, strives to process the sensory data in the most reactive way possible, without sacrificing its global convergence guarantee. The algorithm uses two modes of motion, called motion-toward-the-target and obstacle-surface-traversal. During the first mode of motion, the robot follows the locally shortest path to the target in a purely reactive fashion. During the second mode of motion, the robot attempts to reach exit points along an obstacle surface, while simultaneously expanding its knowledge of the obstacle based on range data. We provide analysis of the algorithm, showing that if the target is reachable, the robot always finds obstacle exit points from which it reaches the target. Otherwise, the robot eventually possesses full knowledge of the obstacle blocking its path to the target and determines that the target is unreachable. We have also implemented and verified the algorithm on three-dimensional simulated environments. The simulation results show that 3DBug generates paths that resemble the globally shortest path in simple scenarios and reasonably short paths in concave roomlike environments.

The International Journal of Robotics Research
Vol. 20, No. 1, January 2001, pp. 6-25,

1. Introduction

Autonomous robots navigating in a realistic setting must use sensors to perceive the environment and plan accordingly. Sensor-based motion-planning approaches use either global or local planning. In the global planning approach, the robot builds a global model of the environment based on sensory information and uses this model for planning the path (e.g., Choset and Burdick 1995; Cox and Yap 1988; Foux, Heymann, and Bruckstein 1993; Rimon 1997; Stentz 1994).

Global planners are guaranteed to reach the target or conclude target unreachability, but they are difficult to implement due to the inherent uncertainty in sensor data (Crowley and Demazeau 1993). In contrast, local planners are simple to implement since they typically strive to act in a reactive fashion, using navigation vector-fields that directly map the sensor readings to actions. (A planner is called *reactive* if it considers only the currently visible obstacles for planning the next step.) However, to guarantee convergence to the target, the local planners augment their local planning with a globally convergent criterion that influences the robot's local decisions.

A classical family of local planners, called the Bug algorithms, was originated by Lumelsky and Stepanov (1987) and subsequently studied by Kamon, Rimon, and Rivlin (forthcoming); Lumelsky (1991); Noborio and Yoshioka (1993); and Sankaranarayanan and Vidyasagar (1991). These algorithms assume *two-dimensional* configuration spaces, so that each obstacle is bounded by a simple closed curve. In the basic *Bug* algorithm, the robot initially moves toward the target until it hits an obstacle. On that occasion, the robot switches to a boundary-following mode of motion. It leaves the obstacle boundary and resumes its motion toward the target when an *exit condition*, which monitors a globally convergent criterion, holds. If the robot completes a loop around an obstacle without satisfying the exit condition, the robot concludes that the target is unreachable. Our goal is to provide an effective *Bug*-type algorithm for navigation in *three dimensions* (potential applications are discussed below). This is a nontrivial goal, as the obstacle boundaries are now two-dimensional surfaces, while the robot's path is a one-dimensional curve. Consequently, the robot cannot conclude target unreachability by simply completing a loop around an obstacle. Rather, the robot must check an exit condition on the *entire surface* of the obstacle blocking its way to the target before concluding target unreachability. Furthermore, this check must be carefully integrated with the navigation task, to allow efficient departure from an obstacle when the target is reachable.

We assume a three-dimensional polyhedral environment and present two results for sensor-based navigation in this environment. The first result establishes that a robot equipped with a range sensor can visually explore the entire surface of a polyhedral obstacle by tracing only the convex edges within the obstacle's convex hull. This collection of edges is *minimal*, and it is analogous to the *tangent graph* used for navigation in two dimensions (Liu and Arimoto 1992; Rohnert 1986). The second result establishes that a sensor-based estimation of the shortest path to the target can be based on a structure called the *blocking contour*, in time that is linear with the problem size. For comparison, the exact computation of the globally shortest path in a completely known polyhedral space is NP-hard (Canny 1988). Moreover, existing approximation algorithms compute an ϵ-approximate shortest path in a polynomial time of $O(\frac{n^3}{\epsilon^2})$ (Choi, Sellen, and Yap 1994; Papadimitriou 1985), where n is the total number of edges. Thus, our $O(n)$ approach for estimating the shortest path is an attractive alternative that is suitable for reactive navigation in three dimensions.

We incorporate the two results into a globally convergent navigation algorithm called *3DBug*. The algorithm navigates a bug robot in a general three-dimensional polyhedral environment using position and range sensors. The algorithm falls within the framework of the *Bug* algorithms as it strives to process the sensory data in the most reactive way possible, without sacrificing the global convergence guarantee. The algorithm uses two modes of motion, called motion-toward-the-target and obstacle-surface-traversal. During the first mode of motion, the robot follows the locally shortest path to the target in a purely reactive fashion. During the second mode of motion, the robot attempts to reach exit points along an obstacle surface, while simultaneously expanding its knowledge of the obstacle based on range data. If the target is reachable, the robot always finds an exit point and resumes its motion toward the target. Otherwise, the robot eventually possesses full knowledge of the obstacle surface blocking its path to the target and determines that the target is unreachable.

In the related literature, Kutulakos, Lumelsky, and Dyer (1993) suggest a scheme for sensor-based three-dimensional navigation, which combines a two-dimensional *Bug* algorithm with three-dimensional surface exploration. Much like our approach, they argue that the reactive behavior of the two-dimensional algorithms must be relaxed during three-dimensional obstacle exploration to guarantee algorithm completeness. However, in their algorithm, the motion toward the target and the convergence mechanism are restricted to a plane, while in *3DBug*, the entire motion is fully three dimensional. Furthermore, no implementation of their algorithm has been reported, while *3DBug* has been implemented and validated in three-dimensional simulated environments. The problem of sensor-based surface exploration is also discussed in the context of map-building tasks (Kutulakos, Dyer, and Lumelsky 1994; Rao et al. 1988). However, map-building tasks are concerned with efficient exploration of *all* obstacles in the environment. In contrast, navigation tasks focus on efficient navigation to a given target, and surface exploration is only a means for circumnavigating obstacles during this motion. Other works consider visual exploration in the context of grasping applications (Blake, Zisserman, and Cipolla 1992; Connoly 1985; March and Chaumette 1997; Maver and Bajcsy 1993; Whaite and Ferrie 1991). However, these works are not directly relevant to the problem of online navigation.

Effective sensor-based navigation algorithms in three dimensions have several potential applications. First, they can be useful for planning the motion of robot manipulators designed to operate in complex unstructured environments (e.g., Crane, Duffy, and Carnahan 1991; TITAN 1996). One can install a three-dimensional sensor device such as the "eye" sensors of Nayar (1997) and Svoboda, Pajdla, and Hlavac (1998) near the end effector, and use an algorithm such as *3DBug* to guide the motion of the end effector without any a priori knowledge of the environment. (In this context, the links of the mechanism must be further protected from collision.) A second potential use of algorithms such as *3DBug* is for *off-line* virtual reality applications. For example, movie scenes where high-speed vehicles chase each other in a congested environment can be generated by motion-planning algorithms that use vehicle-based sensor readings. Last, *3DBug* is a natural stepping stone toward a longer-term goal of achieving reactive sensor-based

navigation in the (x, y, θ) configuration space of mobile robots. Current reactive sensor-based implementations for mobile robots approximate the shape of the vehicle by a cylinder and use 2-degree-of-freedom algorithms to navigate the robot (Chatila 1995; Laubach, Burdick, and Matthies 1998). The ability to include the robot's rotational degree of freedom in the planning can significantly enhance the mobile robot maneuverability.

The paper is organized as follows. We first present two basic results useful for sensor-based navigation in three dimensions. Then, we present and analyze the properties of the *3DBug* algorithm. Next, we describe simulation results, showing that *3DBug* generates paths that often resemble the globally shortest path to the goal in simple environments and reasonably short paths in concave roomlike environments. The concluding section outlines future work and extensions of *3DBug*.

2. Basic Results for Navigation in Three Dimensions

In this section, we characterize the sensor data and then discuss two basic results useful for sensor-based navigation in three dimensions. First, we show that the robot can visually explore the entire surface of a polyhedral obstacle by tracing the convex obstacle edges in its convex hull. Then, we show that the locally shortest path is determined by a structure called the blocking contour, and we use this structure to formulate an efficient method for estimating this path.

2.1. The Sensor Data

To begin with, we assume a position sensor that measures the robot's current location, denoted X. We also assume a range sensor with an *infinite* detection range, which provides perfect readings of the distance from the robot to the obstacles. The distance readings are acquired only for obstacle points in the *visible set*, which is the three-dimensional star-shaped set centered at the robot's current location X. We further assume that the range data are transformed into the three-dimensional coordinates of all the vertices and edges of the visible obstacles. The assumption of infinite detection range is justified in most indoor environments, as today's range sensors (such as laser scanners) measure distances to objects within a range of at least 50 meters. A similar assumption has been made by other researchers, e.g., Rao et al. (1988).

In a polyhedral environment, the obstacles' visible surfaces are planar polygons (Rieger 1990). To gain insight into the possible arrangements of the visible surfaces, let us mention some relevant results of singularity theory (Whitney 1955). First, singularity theory guarantees that the contours that bound the visible surfaces are topologically stable under small perturbations in generic viewing positions. Second, the theory guarantees that only the following three kinds of generic singularities generate the contours of the visible surfaces (Fig. 1). The first singularity is an *occluding contour*, which is generated at surface points where the viewing direction is perpendicular to the surface normal. This is the only type of singularity that generates whole curves, and in polyhedral environments these curves are visible *convex edges*.[1] The second type of stable singularity is the *T-junction*, which appears as a single point, at a point where an occluding contour meets another occluding contour and becomes invisible. The third singularity, called a *cusp*, also appears as a single point, at a point where a visible occluding contour terminates. We may therefore conclude that each visible surface is bounded by edges of two types—convex obstacle edges and edges generated from occlusion by convex edges of other obstacles (see Fig. 1).

2.2. Sensor-Based Surface Exploration

Surface exploration is a key component in the process of circumnavigating a three-dimensional obstacle. We now show that the entire surface of a polyhedron $\mathcal{B}$ can be visually explored while tracing only the convex edges in its convex hull. In general, the boundary of $\mathcal{B}$ may consist of several connected surfaces. We focus on the connected surface of $\mathcal{B}$, which lies in the connected component of $\mathcal{R}^3$ that contains the robot. Let $Co(\mathcal{B})$ denote the convex hull of $\mathcal{B}$, and let $\mathcal{E}$ denote the set of convex obstacle edges that intersect $Co(\mathcal{B})$. Note that some convex edges in $\mathcal{E}$ may belong to neighboring obstacles.

To demonstrate the concept of visual exploration by tracing convex edges, consider the following analogous two-dimensional example. To visually explore the entire outer boundary of a polygon, it suffices to visit all the convex vertices in the polygon's convex hull. For example, the entire boundary of the polygon $\mathcal{B}_1$ in Figure 2(a) is visible from the convex vertices $V_1, \ldots, V_6$ contained in the convex hull of $\mathcal{B}_1$. Moreover, it can be verified that this set is the *minimal* set of vertices, which guarantees visual coverage, since removing a single vertex from this set would destroy the coverage property. For example, removing the vertex V in Figure 2(b) destroys the coverage property. To characterize visibility in three dimensions, we need the following lemma, which is proved in Appendix B.

LEMMA 1. Shortest path in three dimensions. The shortest path in a three-dimensional polyhedral environment is piecewise linear, and the path's vertices lie only on convex obstacle edges.

We can now state a key result concerning visual exploration in three dimensions.

1. An obstacle edge is *convex* if there exists a plane that passes through the edge such that the obstacle locally lies in one halfspace only. An edge is *concave* if it is not convex.

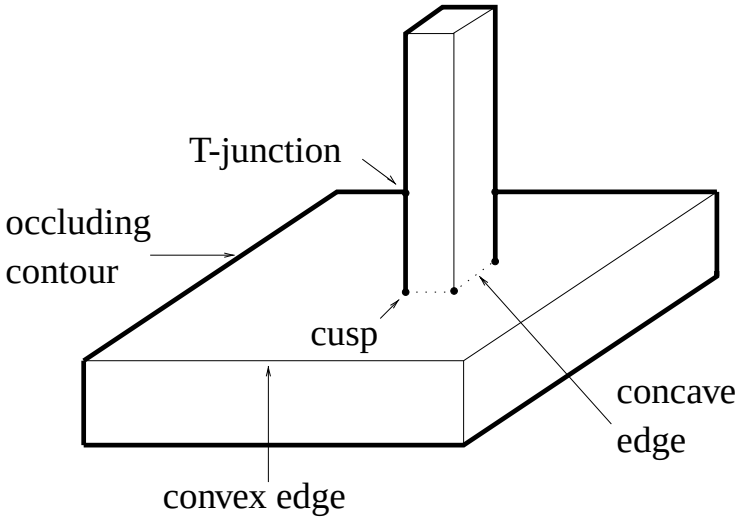


Fig. 1. Examples of convex and concave obstacle edges, and the three types of stable singularities: occluding contour, T-junction, and cusp.

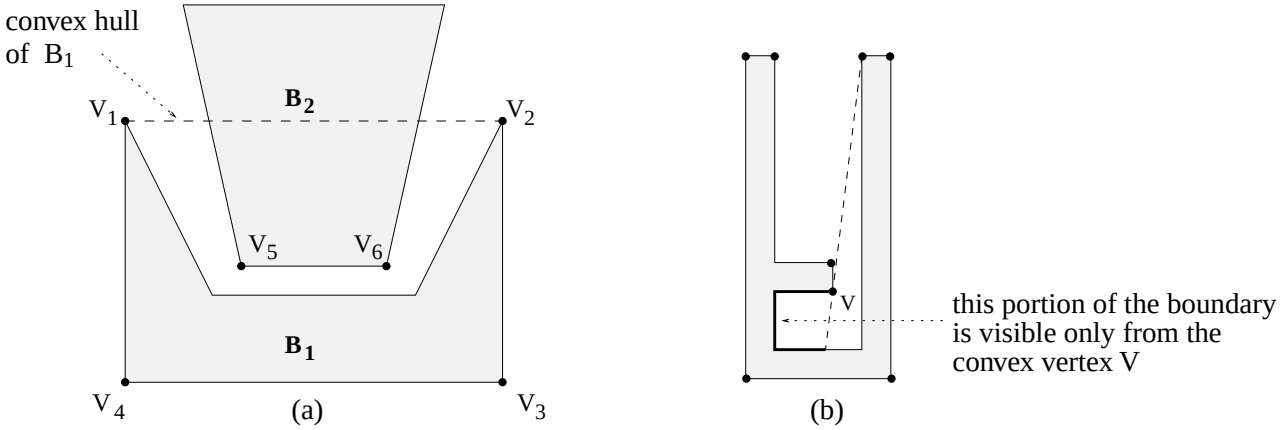


Fig. 2. (a) Visual boundary exploration in two dimensions. (b) All convex vertices are required for visual boundary coverage in two dimensions.

THEOREM 1. Let $\mathcal{B}$ be a polyhedral obstacle in a three-dimensional polyhedral environment, and let $\mathcal{E}$ be the set of convex obstacle edges in $Co(\mathcal{B})$. Then every point of $\mathcal{B}$ is directly visible from some convex obstacle edge in $\mathcal{E}$. Consequently, the entire surface of $\mathcal{B}$ is visible from the convex obstacle edges in $\mathcal{E}$.

Proof. Let $\mathcal{F}'$ be the free space inside $Co(\mathcal{B})$. (In other words, $\mathcal{F}'$ is the complement of the obstacles' interior in $Co(\mathcal{B})$.) We need the following facts concerning $\mathcal{F}'$. First, $\mathcal{F}'$ has polyhedral boundaries since the convex hull of a polyhedron is also a polyhedron. Second, the convex edges in $\mathcal{F}'$ necessarily belong to $\mathcal{E}$. Third, by definition, the surface of $\mathcal{B}$ lies entirely in $\mathcal{F}'$. Since the surface of $\mathcal{B}$ is connected, it is possible to find a path between any two points on the surface of $\mathcal{B}$, such that the path lies in $\mathcal{F}'$.

Let $\mathsf{bdy}(\mathcal{B})$ denote the boundary of $\mathcal{B}$. For a given point $p \in \mathsf{bdy}(\mathcal{B})$, there exists some other point $p_0 \in \mathsf{bdy}(\mathcal{B})$ from which p is *not* directly visible. Consider the shortest path from p_0 to p in $\mathcal{F}'$. According to Lemma 1, this path is piecewise linear and its vertices lie on convex edges in $\mathcal{F}'$. Let $p_0 = q_1, q_2, ...q_{k-1}, q_k = p$ denote the vertices of this path. By construction, p is directly visible from the path vertex q_{k-1}. But q_{k-1} belongs to some convex edge in $\mathcal{F}'$. Since all the convex edges in $\mathcal{F}'$ belong to $\mathcal{E}$, p is directly visible from some convex edge in $\mathcal{E}$. □

The theorem implies that a robot equipped with a range sensor can explore the entire surface of a polyhedral obstacle $\mathcal{B}$ by tracing the convex obstacle edges in $\mathcal{E}$. Furthermore, it is not hard to see that $\mathcal{E}$ is the *minimal* collection of edges that guarantees visual coverage, since a removal of a single edge from $\mathcal{E}$ can destroy the coverage property. Last, a shortest-path argument can be used to show that the convex obstacle edges in $\mathcal{E}$ are "visually connected." That is, for every pair of edges E', E'' in $\mathcal{E}$, there exists a chain of edges in $\mathcal{E}$, $E' = E_1, E_2, \ldots, E_k = E''$, such that E_{i+1} is visible from E_i. It follows that the robot can incrementally trace the entire set $\mathcal{E}$ using range data.

To support surface exploration that is based on the set $\mathcal{E}$, we define in Appendix A a compact data structure called the *convex edges graph* or CEG. The CEG is designed to support two tasks: obstacle surface traversal and computation of shortest paths to the target. The CEG nodes represent convex obstacle edges in $Co(\mathcal{B})$ that have been seen by the robot during the exploration process. The CEG edges connect between CEG nodes in a way that guarantees global connectivity of the CEG. The CEG also contains a special type of edges called *target edges*, which connect each of the CEG nodes to the target. Each target edge emanates from the point closest to the target along the convex obstacle edge corresponding to the CEG node. The weight of each target edge reflects an estimate of the path length from the corresponding obstacle edge to the target, and this weight plays an important role in the ensuing algorithm. The full details of the CEG data structure appear in Appendix A.

2.3. Locally Shortest Path in Three Dimensions

A reactive navigation algorithm plans its path based on the currently sensed obstacles, and we now consider the computation of a locally optimal path based on these data. Let T denote the target. We define the *locally shortest path* from the robot current location X to T as the shortest collision-free path, based only on the currently visible obstacles. Before describing a technique for estimating this path, we introduce some terminology. We model each visible obstacle surface as a polyhedral two-sided thin wall (or shell) in the physical world. If the target is not directly visible from X, there is some *blocking obstacle* between the robot and the target. In this case, the line segment $[X, T]$ crosses the blocking obstacle, and we refer to the visible surface that blocks the segment $[X, T]$ as the *blocking surface*. The blocking surface is bounded by a piecewise linear curve, termed the *blocking contour* (Fig. 3(a)). By construction, the blocking contour lies on the blocking obstacle and its edges are of two types—occluding and occluded. Occluding (or silhouette) edges are convex edges that belong to the blocking obstacle. Occluded edges are line segments generated from occlusion by some other convex obstacle edges that partially occlude the blocking obstacle (Fig. 3(b)).

The following proposition asserts that the locally shortest path always passes through the blocking contour. (In contrast, the globally shortest path does not necessarily satisfy this property.) A proof of the proposition appears in Kamon (1997).

PROPOSITION 1. In a three-dimensional polyhedral environment, let a blocking obstacle lie between the robot location X and the target T. Then, the shortest path from X to T, *considering only the currently visible obstacles*, must pass through the blocking contour.

In principal, it is possible to compute the locally shortest path using ϵ-optimal algorithms (Choi, Sellen, and Yap 1994; Papadimitriou 1985) on the thin-wall model of the currently visible obstacles. But these algorithms are computationally intensive. We now present an alternative method for estimating the locally shortest path by a curve called the *blocking-contour path*. For each point y on the blocking contour, consider a path consisting of two line segments: the visible part $[X, y]$ and the (optimistically) expected part $[y, T]$. The length of this path is $L_{block}(y) = |X - y| + |y - T|$. For each line segment l_i of the blocking contour, we compute the point v_i, which minimizes $L_{block}(y)$. This computation can be done in constant time per line segment l_i. Then we construct a local graph, called the *blocking-contour graph*, consisting of edges from X to each v_i, and (optimistic) edges from each v_i to T (Fig. 4). The blocking-contour path is the shortest path on

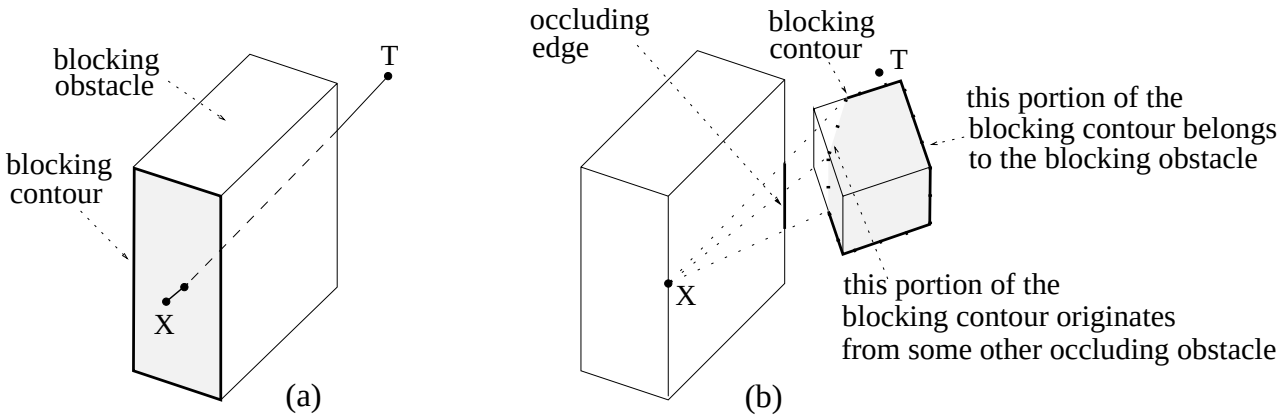


Fig. 3. (a) A blocking contour consisting of silhouette edges only. (b) A blocking contour consisting of both silhouette edges and edges generated from occlusion.

the blocking-contour graph, and it can be found in time *linear* with the number of line segments in the blocking contour.

We may ask, What is the relation of the blocking-contour path to the exact locally shortest path? If the blocking obstacle is convex, the blocking-contour path is precisely the locally shortest path for the following reason. Let y be a point on the blocking contour. If the line segment $[y, T]$ intersects the blocking surface at some interior point, the blocking obstacle must have a visible concavity. Hence, if the obstacle is convex, the line segment $[y, T]$ lies wholly in the free space defined by the thin-wall model of the currently visible obstacles. In particular, this property holds for the blocking-contour path that minimizes the path length $L_{block}(y)$. Consequently, the blocking-contour path is the locally shortest path. However, in general the blocking-contour path is merely an optimistic estimate of the path from X to T. Note that to compute the blocking-contour path, knowledge of the blocking contour is sufficient, and there is no need to construct a full polyhedral model of the blocking surface.

3. The *3DBug* Algorithm

The *3DBug* algorithm navigates a point robot in a three-dimensional unknown environment populated by stationary polyhedral obstacles. The sensory information available to the robot consists of the robot's current position X and range data from X to every obstacle point within the currently visible set. We assume that this sensory information is generated by ideal measuring devices and do not consider here practical issues such as sensor selection, sensor noise, and sensor fusion. First we describe the global structure of the algorithm and then discuss its detailed operation.

3.1. Algorithm Description

The *3DBug* algorithm uses two basic motion-modes: motion toward the target and obstacle surface traversal. During motion toward the target, the robot moves along the locally shortest path based on the currently observable obstacles. At each step of this motion, the robot senses the environment and chooses an intermediate target called *focus point* F. The robot then moves to F without performing any sensing or replanning until it reaches F. While the focus point can be computed continuously during the robot motion, our experience shows that computation of the focus point at discrete steps reduces the computation time without significantly sacrificing the quality of the resulting path. The robot terminates its motion toward the target as follows. Let the *free space* $\mathcal{F}$ be the complement of the obstacles' interiors in $\Re^3$. Let the function $d(w, T) : \mathcal{F} \rightarrow \Re$ measure the Euclidean distance of a point w in the free space from T. The robot keeps moving toward the target until it becomes trapped in the basin of attraction of a local minimum of $d(w, T)$. The appearance of a local minimum is always associated with the presence of a *blocking obstacle*, which blocks the direct path from the robot to the target. At this point, the robot switches to traversing the surface of the blocking obstacle.

During the surface-traversal mode of motion, the robot searches for a suitable exit point on the obstacle surface from which it can resume its motion toward the target. At the same time, the robot expands its knowledge of the obstacle surface and stores this information in the CEG. At each step during surface traversal, the robot computes the *shortest path* to the target based on the current CEG, chooses a focus point F on this path, and moves to F. Upon reaching F, the robot acts in one of the following two ways, according to a regime described below. Either the robot senses the environment, updates the CEG, and immediately moves to a new focus point, or the robot first traces the convex edge containing F while sensing the environment and then updates the CEG and moves to a new focus point. In the edge-tracing mode of operation, the accumulative effect of tracing the convex obstacle edges is the visual coverage of the entire obstacle surface. At the end of each CEG updating, the robot records

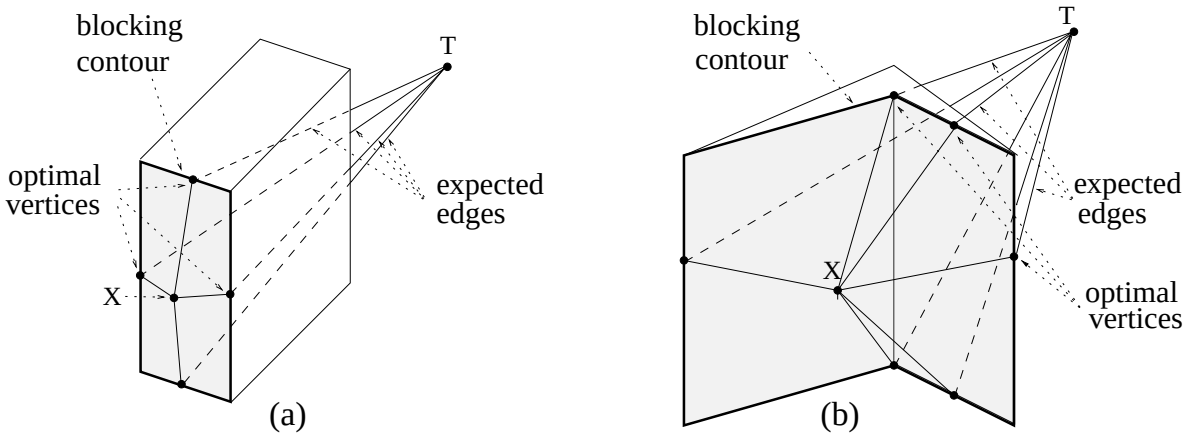


Fig. 4. The blocking-contour graph of (a) a convex obstacle and (b) a concave obstacle.

the closest point to the target observed so far on the obstacle surface. This point is denoted p_{min}.

After updating the CEG, the robot tests an *exit condition* as follows. Let V_{exit} be the closest point to the target along the visible portion of the line segment $[X, T]$, where X is the current robot location. The robot leaves the obstacle surface when $d(V_{exit}, T) < d(p_{min}, T)$. If the robot is unable to find an exit point during the obstacle surface exploration, it performs a final target-reachability test, which is described below. The robot determines that the target is unreachable and halts only if this final test fails. A summary of the algorithm follows.

1. Move toward T along the locally shortest path, until one of the following events occurs:
   - The target is **reached**. Stop.
   - A local minimum is detected. Go to step 2.
2. Traverse the blocking obstacle surface, searching for a suitable exit point while updating the CEG and recording p_{min}, until one of the following events occurs:
   - The target is **reached**. Stop.
   - The exit condition holds: $d(V_{exit}, T) < d(p_{min}, T)$. Go to step 4.
   - The entire surface has been sensed. Go to step 3.
3. Perform the final target-reachability test: go to p_{min}. If the exit condition holds at p_{min}, go to step 4. Otherwise, the target is **unreachable**. Stop.
4. Perform a transition phase. Move directly toward V_{exit} until reaching a point Z where $d(Z, T) < d(p_{min}, T)$. Go to step 1.

3.2. Execution Example

In the following, we present a detailed example of the *3DBug* execution. The environment consists of a single box-like obstacle with a single entry hole (Fig. 5). The start point S is located outside the box such that the entry hole is invisible from S, while the target T is located inside the box. Thus, the robot is forced to explore the side-facets of the box using the surface-traversal mode of motion before it finds the entry hole and reaches T. The robot initially uses the motion-toward-the-target mode of motion. From S, the robot observes the blocking contour, which is the boundary of the visible surface that blocks the robot's direct path to the target (Fig. 5(a)). Next, the robot computes the locally shortest path to the target. This path passes through the blocking contour, and the robot moves to the point F_1 where the locally shortest path intersects the blocking contour. At this stage, the robot moves toward the upper facet of the box, since this facet is not visible from S and is thus unknown to the robot. When the robot reaches F_1, it detects that it cannot further decrease its distance to the target as follows. The robot detects that all the points y on the blocking contour, as defined from F_1, satisfy $d(y, T) > d(F_1, T)$. The robot subsequently switches to the obstacle-surface traversal mode of motion. In the new mode of motion, the robot first constructs the CEG, which contains all the convex obstacle edges visible from F_1. The CEG also contains target edges, which connect its nodes to the target. The robot next computes the shortest path to T along the CEG and moves to the point F_2, which lies on the shortest path (Fig. 5(b)). When the robot reaches F_2, it observes another side-facet of the box. The robot updates the CEG and computes the shortest path to T on the updated CEG (Fig. 5(c)). Note that the CEG has been created at F_1 and not at S. Hence, the edges of the side-facet that contain the point F_2 are added to the CEG only at F_2, although this facet was first seen from S. The shortest path from F_2 to T leads to F_3. As the robot reaches F_3, it observes the entry hole and updates the CEG accordingly (Fig. 5(d)). At this point, the convex obstacle

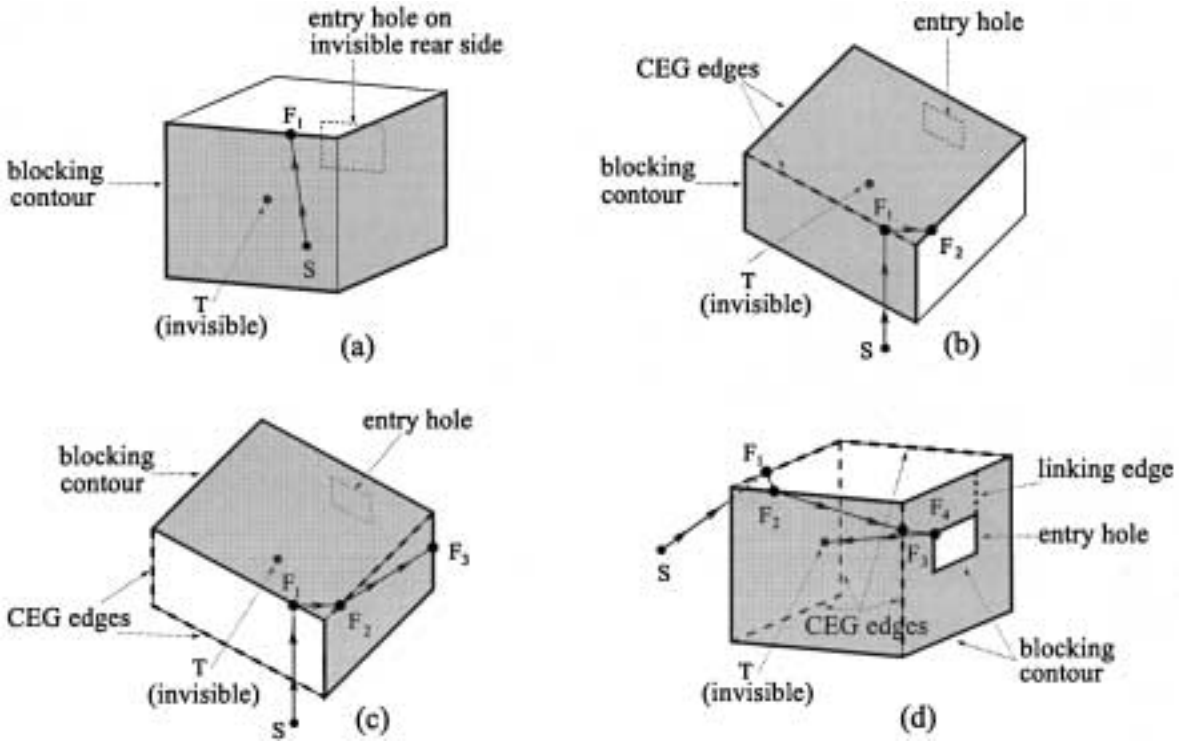


Fig. 5. Execution example of *3DBug*. The figures present the local views from (a) the start point S, (b) the focal point F_1, (c) F_2, and (d) F_3. The visible surfaces from each viewing position are filled.

edges in the CEG form two disjoint components. To make the CEG a connected graph, a linking edge is added to the CEG according to a procedure specified in Appendix A. The robot now computes the shortest path to T along the updated CEG and moves along this path to F_4. At F_4, the target is directly visible and the robot moves directly to T. We now proceed to describe the detailed operation of the algorithm.

3.3. Motion Toward the Target

During motion toward the target, the robot moves between successive focus points along the locally shortest path to the target, based on the currently sensed obstacles. If the target T is directly visible to the robot, the shortest path leads directly to T. Otherwise, the locally shortest path passes through the blocking contour (Proposition 1). To guarantee convergence to the target, we wish to ensure that the distance of the robot to the target decreases monotonically between successive focus points. To achieve this objective, the algorithm computes the locally shortest path based only on the points y of the blocking contour satisfying $d(y, T) \leq d(X, T)$, where X is the current robot location. This subset of the blocking contour is termed the *feasible subcontour* (Fig. 6(a)). Once the feasible subcontour is computed, the algorithm constructs the blocking-contour graph based on the feasible subcontour and the target node and searches this graph for the shortest path to T.

The robot chooses a new focus point F along the locally shortest path as follows. Let Y be the point on the feasible subcontour through which the locally shortest path passes. (It can be verified that Y is unique.) If Y lies on a convex edge of the blocking obstacle, F is set to Y (Fig. 6(a)). If Y lies on a line segment generated by occlusion, F is chosen on the occluding obstacle edge at the point where the line segment $[X, Y]$ crosses the occluding edge (Fig. 6(b)). The reason for this choice is as follows. The globally shortest path never passes through line segments generated by occlusion. Since we wish to achieve local decisions that resemble the globally optimal ones, F is chosen on the occluding edge. In Kamon (1997), we describe a postprocessing step that removes cases in which the decrease in $d(X, T)$ between successive focus points is infinitesimally small. This postprocessing step ensures that the number of focus points in each motion-toward-the-target segment is finite.

The robot terminates its motion toward the target and switches to obstacle surface traversal when it detects that it is trapped in the basin of attraction of a local minimum of the distance function $d(w, T)$. The corresponding sensor-based termination condition is that the feasible subcontour becomes empty. As shown below, this event is always associated with the presence of a local minimum of $d(w, T)$.

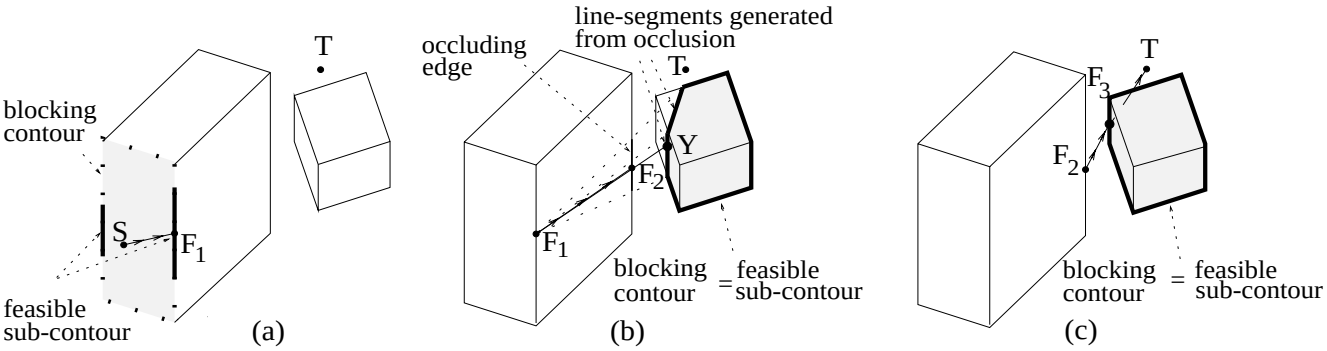


Fig. 6. Motion toward the target. (a) From $X = S$, the locally shortest path passes through F_1. (b) At $X = F_1$, the point Y lies on a line segment generated by occlusion. Hence, F_2 is chosen on the occluding edge. (c) From $X = F_2$, the locally shortest path passes through F_3, from which the target is directly visible.

3.4. Obstacle Surface Traversal

During obstacle surface traversal, the robot attempts to find a suitable exit point while accumulating data about the obstacle surface to determine target unreachability. Let P denote the point where the robot switches to obstacle surface traversal. It can be verified that the local minimum of $d(w, T)$, which terminated the motion toward the target, is visible from P and lies on the surface of the obstacle that blocks the direct path from P to the target (the blocking obstacle). The robot traverses the surface of this obstacle until either an exit condition is satisfied or the entire obstacle surface is explored. Upon starting a new obstacle-surface-traversal motion, the robot moves into the convex hull of the blocking obstacle by choosing a focus point F_0 at the closest point to the target on the blocking contour. Since the blocking contour lies on the blocking obstacle, F_0 lies inside the convex hull of the blocking obstacle. The robot moves directly to F_0, then senses the environment and generates the initial CEG of the blocking obstacle. At each step after the initial one, the robot computes the shortest path to the target along the current CEG. The last edge along this path is always a target edge, which connects a particular CEG node to T. Let V denote this CEG node. Then the robot chooses the next focus point F at the point where the target edge emanates from the convex obstacle edge corresponding to V. The robot next moves to F along the CEG-based shortest path.

Before describing the action taken by the robot at F, let us consider the high-level objectives of the obstacle-surface-traversal mode of motion. This mode of motion has two objectives. The primary objective is to find an obstacle exit point if one exists. The secondary objective is to explore the obstacle surface to conclude target unreachability. Our approach to the integration of these two objectives is based on the following intuitive observation. In polyhedral environments, it often suffices for the robot to sense the environment and test the exit condition from a single point on each convex obstacle edge. In other words, it is often the case that the data collected from a single point on a convex obstacle edge reveal all the information necessary to register neighboring obstacle edges and evaluate the exit condition. This observation suggests that for navigation purposes, the robot need only visit a single point in each convex obstacle edge and then immediately proceed to some other obstacle edge. Only when the collection of detected edges has been checked and no exit point has been found should the robot resort to the relatively time-consuming operation of edge tracing. Based on this argument, we perform the obstacle surface traversal using the following primary and secondary phases.

During the primary phase, the robot checks the exit condition at a single point on each convex obstacle edge. When the robot arrives to a focus point F, it senses the environment and updates the CEG as shown in Figure 7. Next, the robot raises the weight of the target edge that emanates from F to infinity. This weight increase ensures that subsequent CEG paths will lead the robot to other yet unvisited convex obstacle edges. Our experiments have shown that the robot is usually able to effectively find an exit point using only the primary phase of the surface traversal motion. However, if all the detected convex obstacle edges (equivalently, the CEG nodes) have been visited without finding a suitable exit point, the robot reinitializes the target edges to their original weight and executes the secondary phase.

During the secondary phase of obstacle surface traversal, the robot traces entire convex edges. When the robot arrives to a focus point F, it traces the convex obstacle edge, which contains the point F, while continuously sensing the environment and updating the CEG. During this tracing, the robot continuously computes the closest point to the target observed so far on the obstacle surface,[2] p_{min}. At the end of this tracing, the robot tests the exit condition. If the test is not satisfied,

2. To compute p_{min}, the robot decomposes the blocking surface into its constituent planar polygons. For each planar polygon P, the robot computes the closest point to T, denoted y, in the plane of P. If y lies inside P, then $p_{min} = y$. Otherwise, p_{min} is the closest point to T on the boundary of P.

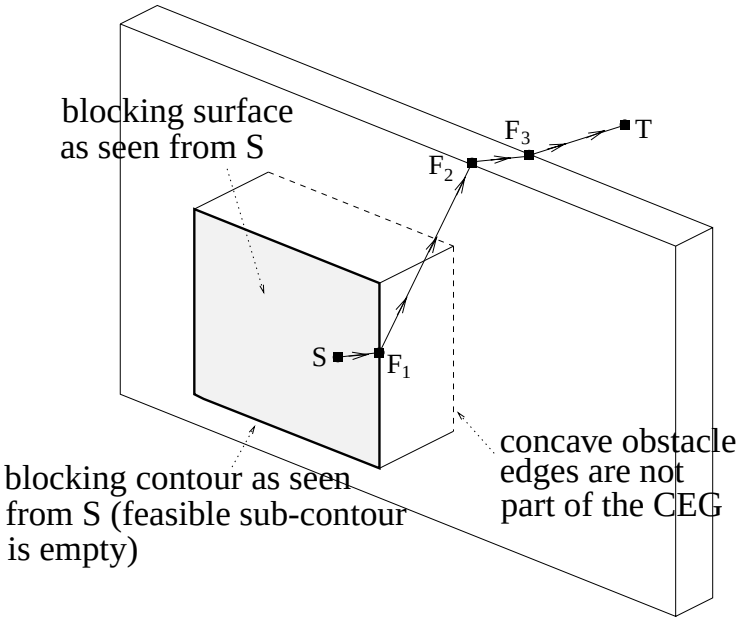


Fig. 7. An example of the primary phase of obstacle surface traversal. The robot switches to surface traversal already at S, since $d(S,T) < d(y,T)$ holds for all points y on the blocking contour. The robot moves through the focus points F_1, F_2, and F_3. From F_3, the target is visible, and the robot leaves the obstacle.

the robot raises the weight of the target edge that emanates from F to infinity. This weight increase ensures that the CEG node corresponding to the traced convex obstacle edge will not be retraced again. However, subsequent CEG paths may pass through this CEG node on their way to yet untraced convex obstacle edges. The secondary phase ends either when the robot finds an exit point or when all the convex obstacle edges corresponding to CEG nodes have been completely traced.

Finally, we discuss the exit condition and the final target-reachability test. After updating the CEG, the robot tests the exit condition by inspecting V_{exit}, the closest point to the target along the visible portion of the segment $[X,T]$. If $d(V_{exit},T) < d(p_{min},T)$, the exit condition is satisfied. Before resuming its motion toward the target, the robot performs a transition phase where it moves directly toward V_{exit} until it reaches a point Z where $d(Z,T) < d(p_{min},T)$. As discussed below, the combination of the exit condition and the transition phase ensures that each local-minimum of $d(w,T)$ is associated with at most one switch to surface traversal. Finally, if the entire surface has been explored without finding an exit point, the robot performs a final target-reachability test. This test is necessary since the exit condition is tested only at discrete points on convex obstacle edges, and these points alone do not suffice to conclusively determine target unreachability. To perform the test, the robot moves to the closest point to the target, p_{min}, and checks the exit condition from there. If the exit condition is not satisfied at p_{min}, the target is unreachable.

4. Algorithm Analysis

The convergence of *3DBug* is based on the following ideas. During motion toward the target, the distance of the robot from the target, $d(X,T)$, decreases monotonically between successive steps. Moreover, the path length of each motion-toward-the-target segment is finite. During obstacle surface traversal, the robot either senses the entire obstacle surface or leaves the surface before completing the exploration. We prove that the path length of each obstacle-surface-traversal segment is finite. The robot switches to surface traversal only at points that are uniquely associated with local minima of the distance function $d(w,T)$. Since $d(w,T)$ has finitely many local minima in any bounded polyhedral environment, there are finitely many motion segments. As each segment is of finite length, the algorithm terminates after a finite-length path. If the target is reachable, convergence to the target is guaranteed by the exit condition. This condition ensures that the robot always terminates its surface traversal mode and resumes its motion toward the target. By construction, the last motion-toward-the-target segment has no obstacle trapping the robot at a local minimum of $d(w,T)$, and this segment leads the robot to T.

Next we introduce some terminology. We consider a point robot in a bounded three-dimensional space, populated by a finite number of stationary polyhedral obstacles. A point where the robot switches from motion toward the target to obstacle surface traversal is termed a *switch point* P_i. Each

switch point P_i has a corresponding *local-minimum point* M_i, which is the local minimum of the distance function $d(w, T)$ that triggers the switch. A point where the exit condition holds and the transition phase is initiated is termed an *exit point* L_i. Last, a *transition point* Z_i is a point where the transition phase terminates and the motion toward the target resumes.

We now summarize the convergence proof of *3DBug*, which appears in full detail in Kamon (1997). The following two lemmas assert that each segment of the algorithm's two modes of motion has finite length.

LEMMA 2. The path length of each motion-toward-the-target segment is finite.

Proof sketch. In the proof, we show that every motion-toward-the-target segment consists of finitely many focus points F_i. By construction, the next focus point F_{i+1} is chosen on a convex obstacle edge that is directly visible from the current focus point F_i. Since F_{i+1} is directly visible from F_i, the robot moves between successive focus points along a straight-line segment of finite length. Hence, the path length of each motion-toward-the-target segment is finite. □

LEMMA 3. The path length of each obstacle-surface-traversal segment is finite.

Proof. Each obstacle-surface-traversal segment has primary and secondary phases. In both phases, the weight of the target edge connecting a convex obstacle edge to T is raised to infinity once the edge has been visited. Thus, the robot visits at most one point in each convex obstacle edge during the primary phase and traces a convex obstacle edge at most once during the secondary phase. Since there are finitely many obstacle edges, both phases take a finite number of steps to complete. Each step involves motion between two nodes along the CEG, together with a possible tracing of the obstacle edge corresponding to the last CEG node along the path. The path between any two CEG nodes has finite length, since by assumption the polyhedral environment is bounded and contains finitely many obstacles. Hence, each step requires motion along a finite-length path, and the total path length of each obstacle-surface-traversal segment is finite. □

We wish to show that there are finitely many segments of each motion mode. We begin with a lemma that associates with every switch point P_i a local minimum of $d(w, T)$. Then we show that the distance to the target decreases monotonically between successive local-minimum points. The proofs of these lemmas appear in Appendix B.

LEMMA 4. Every switch point P_i, where the robot switches from motion toward the target to obstacle surface traversal, has a corresponding unique point M_i, which is a local minimum of $d(w, T)$, such that $d(M_i, T) \leq d(P_i, T)$.

LEMMA 5. The distance to the target decreases monotonically between successive local-minimum points, that is, $d(M_{i+1}, T) < d(M_i, T)$.

The next lemma asserts that if the target is reachable, the exit condition holds true after a finite-length path.

LEMMA 6. If the target is reachable from a switch point P_i, the exit condition will cause the robot to leave the obstacle surface after a finite-length path.

Proof. Since each obstacle-surface-traversal segment is finite in length (Lemma 3), it suffices to show that the exit condition is satisfied in the final target-reachability test, performed at the end of each surface-traversal segment. The final target-reachability test is performed from a point p_{min}, which is the closest to T on the obstacle surface. Since T is reachable, it must be possible to leave the obstacle at p_{min} and move toward T for some, possibly short, line segment that ends at V_{exit}. If T is directly visible from p_{min}, then $V_{exit} = T$. Otherwise, V_{exit} is the point where the line segment $[p_{min}, T]$ crosses some other obstacle, and $d(X, T)$ decreases monotonically along the line segment $[p_{min}, V_{exit}]$. Thus, the exit condition $d(V_{exit}, T) < d(p_{min}, T)$ holds in both cases and the robot leaves the obstacle surface. □

The following theorem asserts that *3DBug* always terminates.

THEOREM 2. (*3DBug* terminates.) The *3DBug* algorithm *terminates* in any three-dimensional polyhedral environment, after following a finite-length path.

Proof. The robot switches to obstacle surface traversal only at points that are associated with unique local-minima of $d(w, T)$ (Lemma 4). The distance to the target decreases between successive local-minimum points (Lemma 5). Thus, each local-minimum point of $d(w, T)$ is associated with at most one switching to obstacle surface traversal. There are finitely many local minima of $d(w, T)$ in any bounded polyhedral space. Hence, the path consists of finitely many obstacle-surface-traversal segments, which are interleaved by motion-toward-the-target segments and transition phases. Lemmas 2 and 3 guarantee that the path length of each motion segment is finite. The path length of each transition phase is finite since the robot moves directly toward a fixed point during this phase. Hence, the total path length generated by the algorithm is finite. □

The following theorem asserts that *3DBug* is complete, that is, that it always finds the target if the target is reachable.

THEOREM 3. (*3DBug* is complete.) The *3DBug* algorithm *finds the target* in any three-dimensional polyhedral environment, provided that the target is reachable from the start point.

Proof. As stated in the proof of Theorem 2, there are finitely many obstacle-surface-traversal segments. If T is reachable from S, Lemma 6 guarantees that every obstacle-surface-traversal segment terminates after a finite-length path. Since every such segment is followed by a transition phase, there is a *last* transition phase. This transition phase is followed by a

last motion-toward-the-target segment, which leads the robot to the target. □

5. Simulation Results

This section describes simulation results that compare the path generated by *3DBug* to the globally shortest path. We present the average performance of *3DBug* in five environments and describe particular examples in detail. Then, we discuss some general search characteristics of *3DBug*.

To simulate the *3DBug* algorithm, we developed a three-dimensional range-sensor simulator, which computes the blocking surface in environments populated by general polyhedra. The simulator is based on the solid modeling package IRIT (Technion—Israel Institute of Technology 1997). Since computing the globally shortest path in a polyhedral environment is NP-hard (Canny 1988), we approximate this path by constructing and searching a three-dimensional generalized visibility graph (Lozano-Perez and Wesley 1979). In this graph, the obstacle edges are broken into fixed-length segments, and each edge-segment becomes a node. The edges of the generalized visibility graph are the collision-free lines that interconnect all edge-segments. For example, Figure 8(b) shows the shortest path computed on the generalized visibility graph using a resolution of 0.1 units. The obstacle in this example has a size of 2×0.6×0.1 units, and a resolution of 0.1 units means a partition of the edges into 20×6×1 segments. As the size of the edge-segments decreases to zero, the shortest path on the generalized visibility graph approaches the exact globally shortest path.

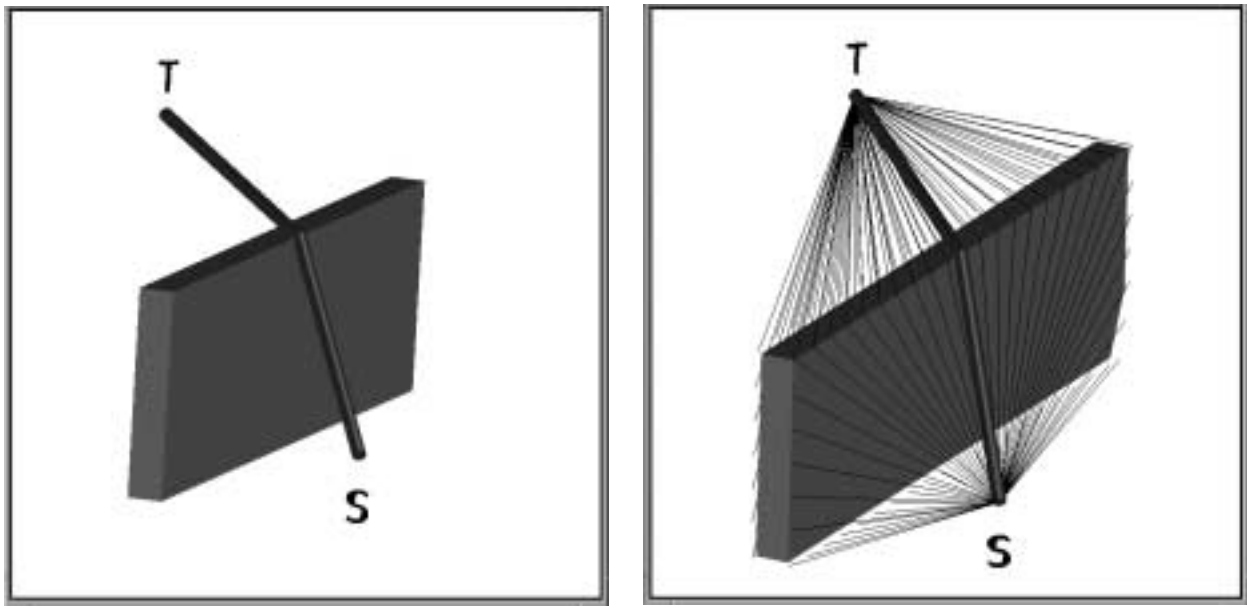
The average results of running *3DBug* on five simulated environments are summarized in Table 1. The algorithm was tested in 400 runs for each environment, with randomly chosen start and target points. The target T was always reachable but was chosen such that it was invisible from the start point S. For comparison, we also computed the globally shortest path on the generalized visibility graph for each run, using a resolution of 0.1 units. The results listed in the table express the ratio between the average length of the paths generated by *3DBug* and the approximate globally shortest paths. The environment *env1* consists of a single boxlike obstacle (Fig. 8). In this environment, the *3DBug* paths were almost identical to the visibility-graph paths in all of the runs. The next environment, *env2*, is more complex and consists of seven boxlike obstacles (Fig. 9). The average path length of *3DBug* in *env2*, relative to the 3D visibility graph shortest path, was 1.02. In both *env1* and *env2*, the algorithm used the motion-toward-the-target mode of motion along the entire path in over 99% of the runs. The behavior of *3DBug* in *env2* is further discussed below.

The environment *env3* consists of a single concave obstacle that resembles a room with a door and a window (Fig. 10). The average path length in this environment was 1.06 (relative to the 3D visibility graph shortest path), and the obstacle-surface-traversal mode of motion was activated in 65% of the runs. It is interesting to note that there was a significant difference between moving out from the room and moving into the room. The average path length while moving out from the room was 1.01, while the average path length of moving into the room was 1.13. This difference can be explained as follows. Moving out from the room is easier for the robot, since the exits from the room (the door and window) are directly visible from start points inside the room. In contrast, when the robot starts outside the room, it is not always able to see the entries to the room and must first search for these entries (Fig. 10). The environment *env4* consists of two roomlike obstacles, separated by a wall (Fig. 11). The start and target points were always placed inside or near the rooms, on different sides of the separating wall. The average path length of 1.03 in *env4* is better than the average path length in *env3*, since the entries to each room are visible as the robot approaches it from the other room. To summarize, the simulation results indicate that the locally shortest path resembles the globally optimal one in simple environments consisting of disjoint convex obstacles. Moreover, the algorithm generates reasonably short paths even in more complex environments, which include concave roomlike obstacles.

In the last experiment, we tested the *3DBug* algorithm under specifically unfavorable conditions. The environment *env5* consists of a closed box with a small hole near one of its corners (Fig. 12). We call the wall that contains the hole the *front wall*, the wall on the opposite side the *rear wall*, and all the other walls *side walls*. As mentioned above, the navigation task is more difficult when the robot moves into the room from outside, such that the entry to the room is not visible from the start point. In our experiment, S was randomly chosen within a $1 \times 1 \times 1$ volume located near the rear wall, such that the front wall was always invisible from S. The target was randomly chosen within the box, whose size is also 1×1×1 units. This construction forces the robot to explore the side walls using the obstacle-surface-traversal mode of motion. Moreover, since the entry hole into the box is small and located near one of the corners, the length of a typical path from the vicinity of the front wall to the target via the hole is relatively long. Thus, both the environment and the choice of start/target locations are unfavorable for *3DBug*. The average path length over 200 runs, in which the robot moves into the room, was 1.67. The average path length over 200 runs in which the robot moves out from the room was 1.03. This example demonstrates how the limited nature of local information can sometimes lead to significantly long paths.

6. Concluding Discussion

We presented basic results in sensor-based surface exploration and locally shortest path computation in three-dimensional polyhedral environments. We showed that the entire surface

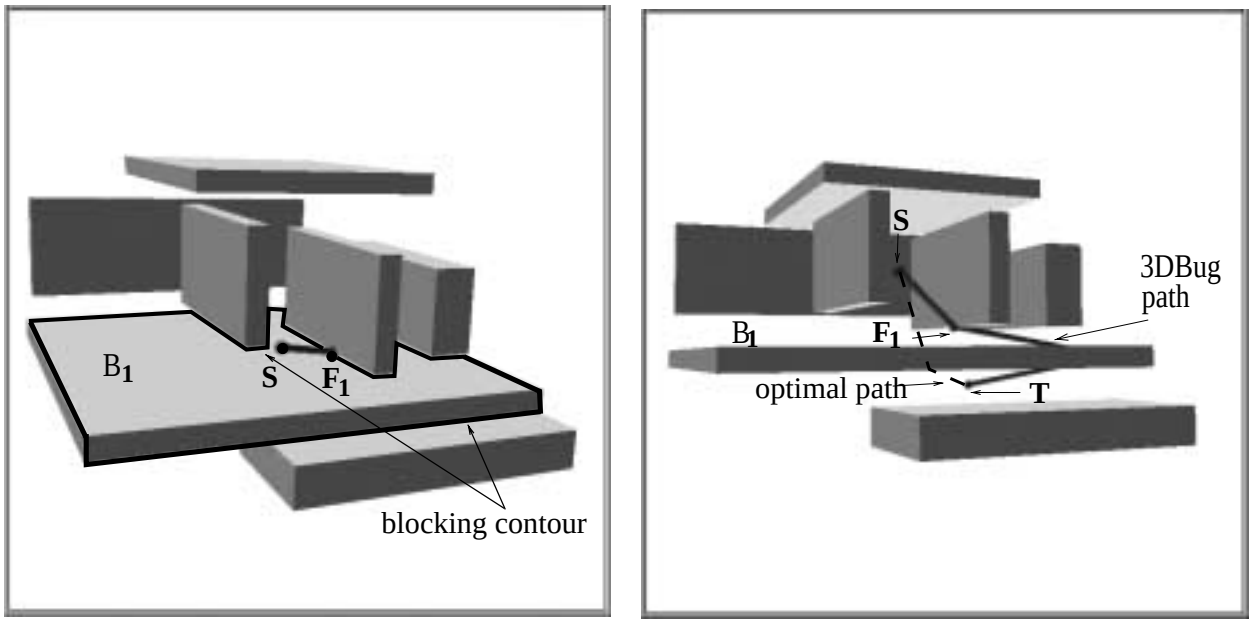


(a) (b)

Fig. 8. A path generated by *3DBug* in *env1*. (b) The 3D visibility graph with resolution 0.1 (only edges which connect S and T to other nodes are presented), with the approximate globally shortest path overlaid as a thick curve.

Table 1. Simulation Results of *3DBug*, Relative to the Approximate Globally Shortest Path

	env1	*env2*	*env3*	*env4*	*env5*
3DBug	1.00	1.02	1.06	1.03	1.35



(a) (b)

Fig. 9. The environment *env2*. (a) The visible surfaces as seen from S. The locally shortest path leads to F_1 since the blocking obstacle $\mathcal{B}_1$ is only partially visible from S. (b) The path generated by *3DBug*, compared to the globally optimal path.

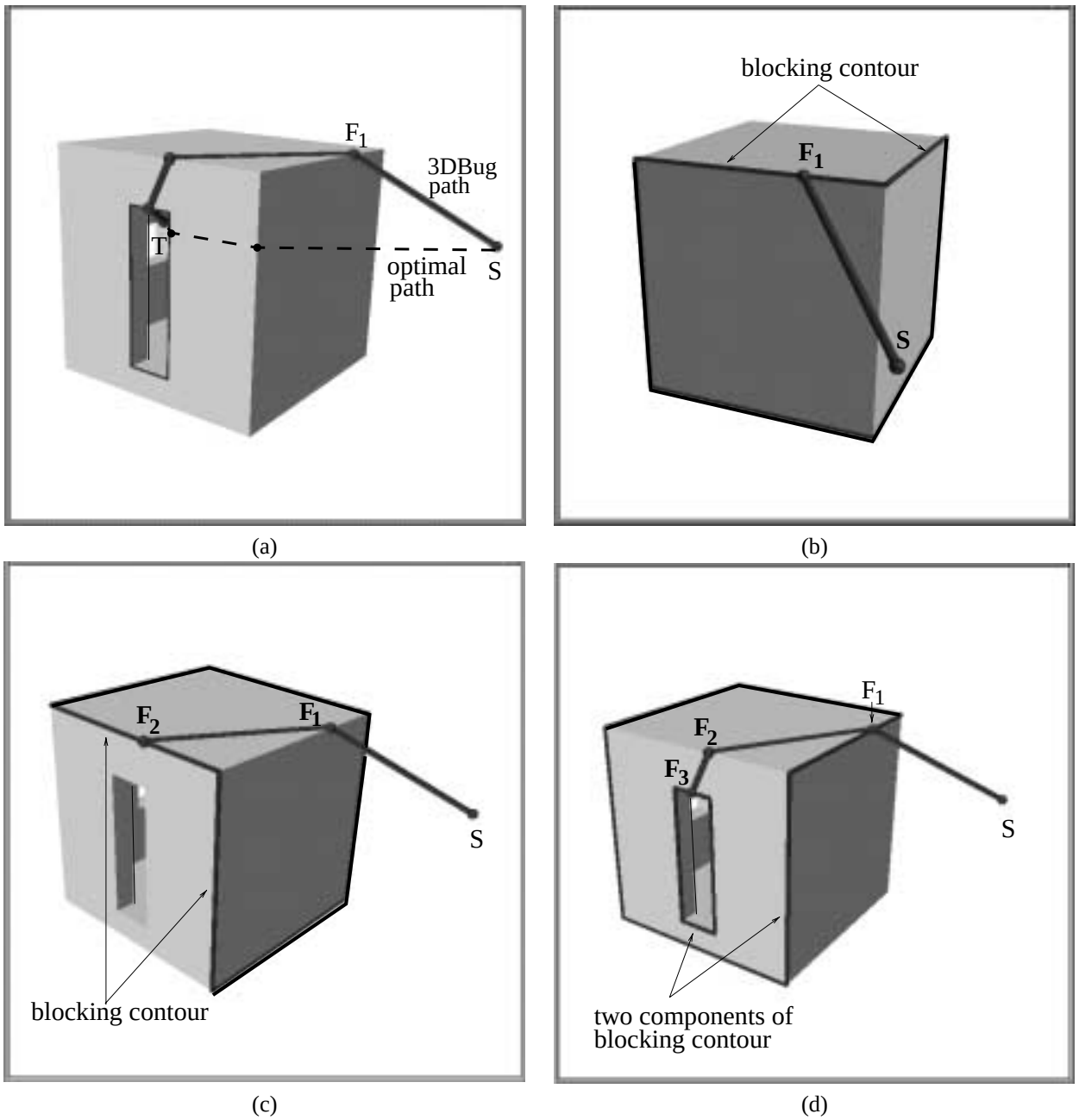


Fig. 10. *3DBug* in the roomlike environment *env3*, as the robot moves into the room. (a) The entire path of *3DBug*, compared to the globally optimal path. The blocking contour shown in bold line, as seen from (b) S, (c) F_1, and (d) F_2. The target is directly visible from F_3.

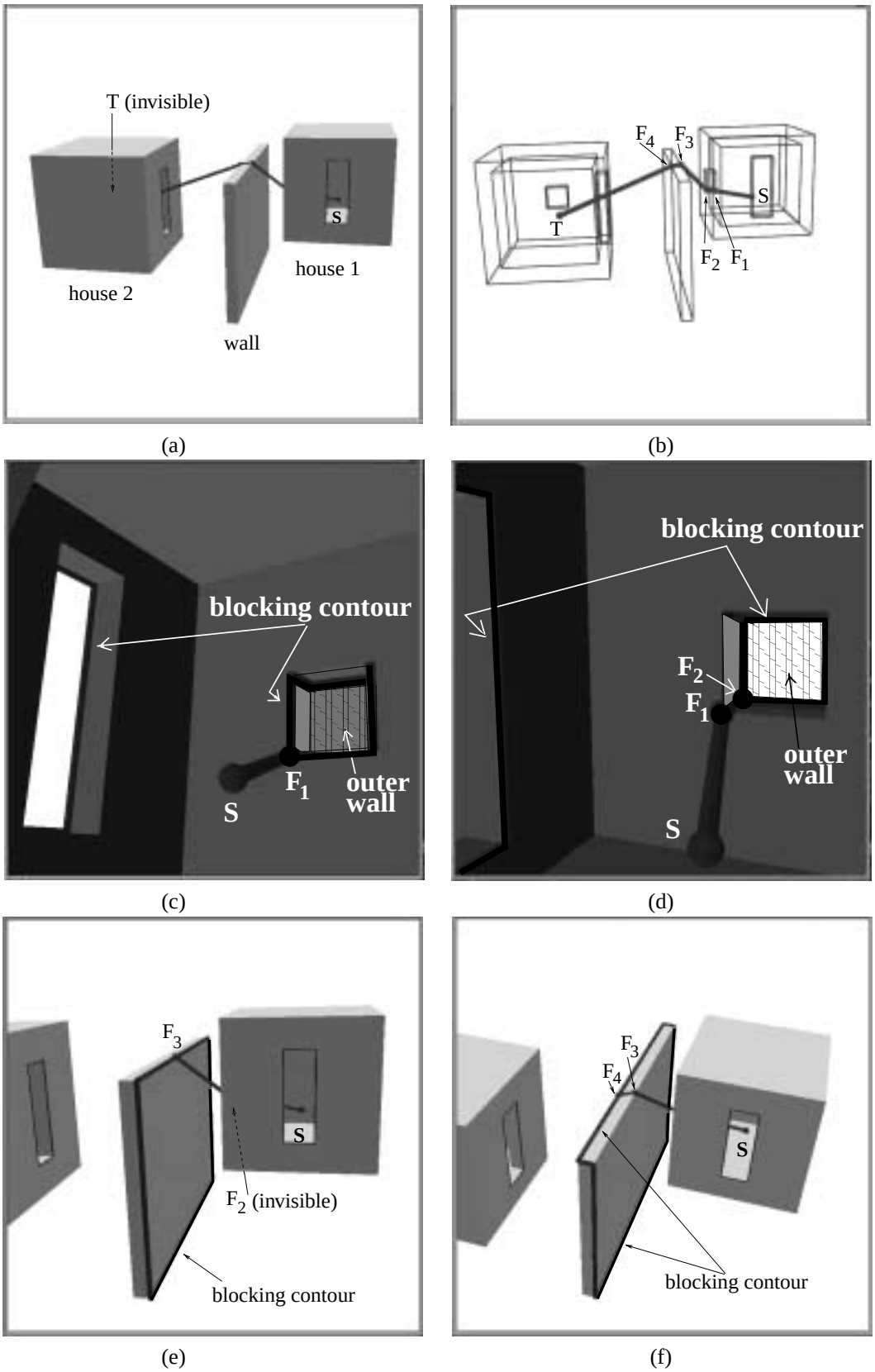
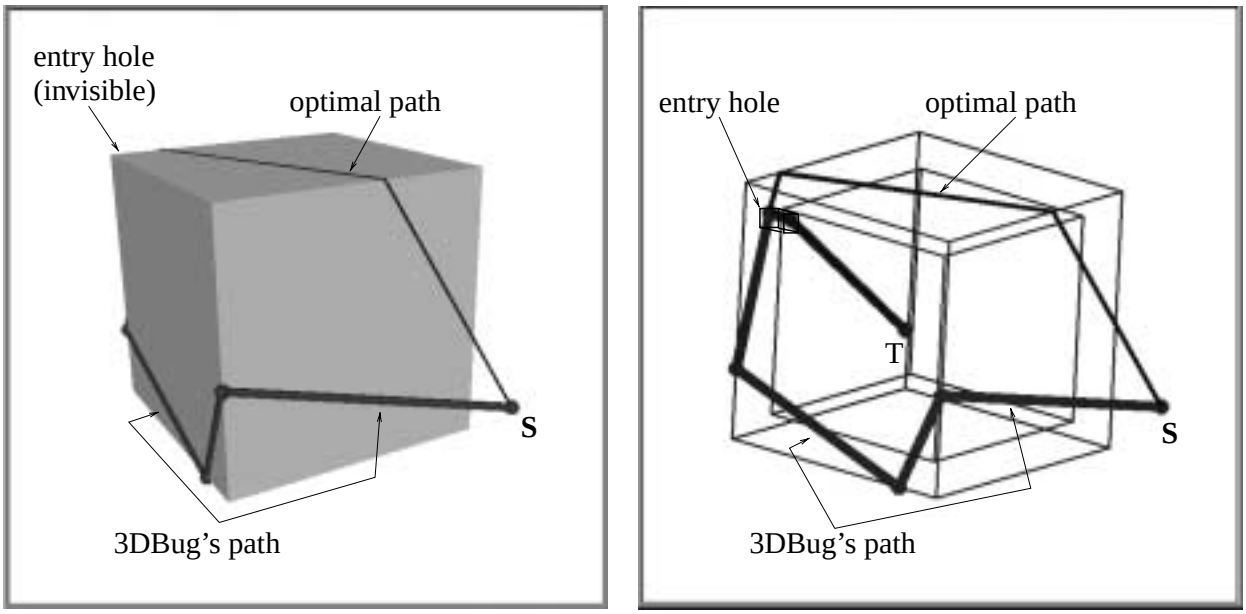


Fig. 11. *3DBug* in *env4*. (a),(b) The robot leaves *house1* through the window and enters *house2* through the door. The globally optimal path is almost identical to *3DBug*'s path. The blocking contour as seen from (c) S, (d) F_1 (located at the internal window frame), (e) F_2 (located at the external window frame of *house1*), and (f) F_3. The target is directly visible from F_4.



(a) (b)

Fig. 12. A particular path generated by *3DBug* in *env5*, compared to the optimal path. The path length is 1.29.

of a polyhedral obstacle can be visually covered by tracing the convex obstacle edges within the obstacle's convex hull. Based on this result, we described a data structure, the *convex edges graph* or CEG, which consists of convex obstacle edges and supports both surface exploration and navigation along an obstacle surface. The CEG is the minimal data structure that supports visual coverage of an obstacle surface and is easy to maintain relative to previously proposed data structures. We also introduced the notion of a locally shortest path in three-dimensions, and we investigated the properties of this path. We showed that the locally shortest path must pass through convex obstacle edges that form the *blocking contour* of an obstacle. Based on this property, we described a technique for efficiently estimating the locally shortest path in time linear with the number of edges in the blocking contour.

These results have been incorporated into a globally convergent algorithm called *3DBug*. The algorithm navigates a point robot equipped with position and range sensors in general polyhedral environments. The algorithm strives to process the sensory data in the most reactive way possible, without sacrificing the global convergence guarantee. During motion toward the target, the robot follows the locally shortest path in a purely reactive fashion. During traversal of an obstacle surface, the robot incrementally constructs the CEG of the obstacle being followed, while attempting to find exit points along the obstacle surface. The analysis of the algorithm confirms that *3DBug* converges to the target in any polyhedral environment. The simulation results show that *3DBug* generates paths that resemble the globally shortest paths in simple scenarios and reasonably short paths in concave roomlike environments.

The *3DBug* algorithm can be used to navigate free-flying robots both in real tasks such as surveillance and simulated tasks such as virtual reality applications. To highlight the advantage of *3DBug* in off-line applications, let us compare *3DBug* with the conventional approach of running a global search algorithm such as A^*. We consider an A^* algorithm, which uses the 3D visibility graph as the underlying search space.[3] *3DBug* finds the target in fewer steps than A^*, since the candidate locations for the next step in *3DBug* are limited to a *single obstacle*, which is the blocking obstacle in both modes of motion. In contrast, A^* must consider *all* nodes that are visible from each node v in the visibility graph. Moreover, restricting the candidate locations of *3DBug* to the blocking contour during motion toward the target is equivalent to considering only tangent edges (i.e., edges that are tangent to obstacles at their endpoints) in the visibility graph. The tangent edges typically constitute a small fraction of the total visibility edges. Thus, *3DBug* processes a smaller amount of information and finds the target in fewer steps than A^*.

We have implemented A^* on the 3D visibility graph and compared its performance to *3DBug*. In *env3*, for example, *3DBug* reached the target after 3.3 steps on average, while A^* required 32.4 steps to reach the target. The advantage of *3DBug* is even more pronounced when the target is unreachable. *3DBug* concludes target unreachability after exploring the entire surface of a single obstacle in which the target is trapped, while A^* must expand all nodes in its search space to conclude unreachability. Another advantage of *3DBug* is

3. A^* is not suitable for physical sensor-based search, since the location of the most promising node, which corresponds to the current robot location, "jumps" discontinuously during the search process.

its efficient memory requirements, since it uses only a limited amount of global information. In contrast, a data structure that represents the entire environment may be very large. For example, the 3D visibility graph of *env2* with resolution 0.1 consists of 620 nodes and 118,912 edges, while *3DBug*'s data structure consists on the average of 7 nodes and 9 edges.

Last, the problem of sensor-based navigation of a point robot in three dimensions is a natural step toward a longer-term goal of sensor-based navigation in the (x, y, θ) configuration space of mobile robots. The extension of *3DBug* to the case of mobile robots is currently under investigation. In the context of this investigation, the following two technical problems arise. The first is how to model the visible set measured by conventional sensors in the physical world as a visible c-space region in the (x, y, θ) configuration space. The second is how to bridge the gap between the polyhedral configuration space assumed for a point robot and the inherently curved configuration space of a mobile robot. The resolution of these and other issues will yield reactive sensor-based algorithms that achieve much greater maneuverability than today's 2-degree-of-freedom algorithms.

Appendix A. The Convex Edges Graph

In this appendix, we define a data structure called the *convex edges graph* or CEG, which supports efficient sensor-based traversal of a polyhedral obstacle $\mathcal{B}$. Before describing the CEG, let us consider the exploration process of $\mathcal{B}$. At each step of the process, the robot moves to a viewing position on some convex obstacle edge in $Co(\mathcal{B})$. Then, the robot traces the obstacle edge while continuously sensing the environment. At the end of this tracing, the robot updates the CEG and moves to a new viewing position on some other convex obstacle edge. We now proceed to define the CEG itself.

First, we describe the CEG nodes. By definition, the CEG nodes represent portions of convex obstacle edges in $Co(\mathcal{B})$ that have been seen by the robot during the exploration. Several CEG nodes may correspond to different portions of a single obstacle edge. These visible portions only expand during exploration, and two CEG nodes may merge into a single node during the exploration. To make the CEG also useful for motion planning, we add the target T as a special node to the CEG. An example of CEG nodes $V_1, \ldots, V_9$ together with T is shown in Figure 13.

Next, we describe the CEG edges. There are three types of CEG edges, each having its weight defined differently. Type 1 edges, termed *point edges*, connect convex obstacle edges, which share a vertex. The weight of such edges is always zero. Type 2 edges, termed *linking edges*, guarantee that the CEG is connected by the end of each exploration step. The linking edges are chosen after the CEG nodes were updated and point edges were added. These edges are selected based on the currently observable obstacles, according to the following three criteria. Linking edges connect disjoint subsets of CEG nodes, they do not intersect any obstacle (i.e., they are visibility edges), and they have minimal length among the edges that satisfy the two previous requirements. It can be verified that it is always possible to find such linking edges that make the CEG a connected graph. The weight of a linking edge is set to its Euclidean length. Note that the endpoints of a linking edge may lie in the interior of an obstacle edge, as illustrated in Figure 13(a). Type 3 edges, termed *target edges*, are abstract edges that connect the CEG nodes to the target T. Each target edge emanates from the point closest to the target along the convex obstacle edge corresponding to the CEG node. The weight of each target edge reflects an estimate of the path length from the corresponding obstacle edge to the target. The path length is computed based on the accumulated data about the obstacle surface, and the details of this computation are described in Kamon (1997). The three types of CEG edges are shown in Figure 13(b).

Finally, we define a notion of path length on the CEG. A path between two CEG nodes is a chain of CEG edges connecting the two nodes. Since a CEG node represents a portion of a convex obstacle edge, two CEG edges may emanate from the same CEG node at different physical points. For example, in Figure 13(a), the edges that connect the CEG node V_5 to the nodes V_2 and V_7 emanate from different points on the convex obstacle edge corresponding to V_5. Hence, to compute the length of a CEG path, we collect not only the weights of the CEG edges along the path but also the length of the edge segment between the entry and exit points within each CEG node along the path. Rather than give a formal definition of this process, let us consider an example. Consider the path from the robot's current location X to T along the CEG shown in Figure 13(a). The robot is located on a convex obstacle edge corresponding to the CEG node V_2. We regard the point X as a special CEG node and connect it to V_2 with a point edge denoted a. The node V_2 is connected to V_5 by another point edge denoted b. Let $l_V(i, j)$ denote the length of the edge segment within the CEG node V between the entry points of the edges i and j. Then, the length of the CEG path from X to T is given by $l(X, T) = L_{V_2}(a, b) + L_{V_5}(b, c) + |c| + |d|$, where $|c|$ and $|d|$ are the weights of the linking edge c and the target edge d.

Let us compare the CEG with two other data structures proposed in the literature for obstacle surface exploration. The first data structure is the complete polyhedral model of an obstacle (Rao et al. 1988). To achieve efficient navigation to the target, both of these structures must support shortest path computation. Consequently, both the CEG and the complete polyhedral model must consider all objects that lie in the obstacle's convex hull, $Co(\mathcal{B})$. Hence, the complete model would contain all the object features in $Co(\mathcal{B})$, while the CEG contains only the convex edges in $Co(\mathcal{B})$. The second alternative data structure is the visible rim (Kutulakos, Lumelsky, and Dyer 1993), which is the collection of curves that

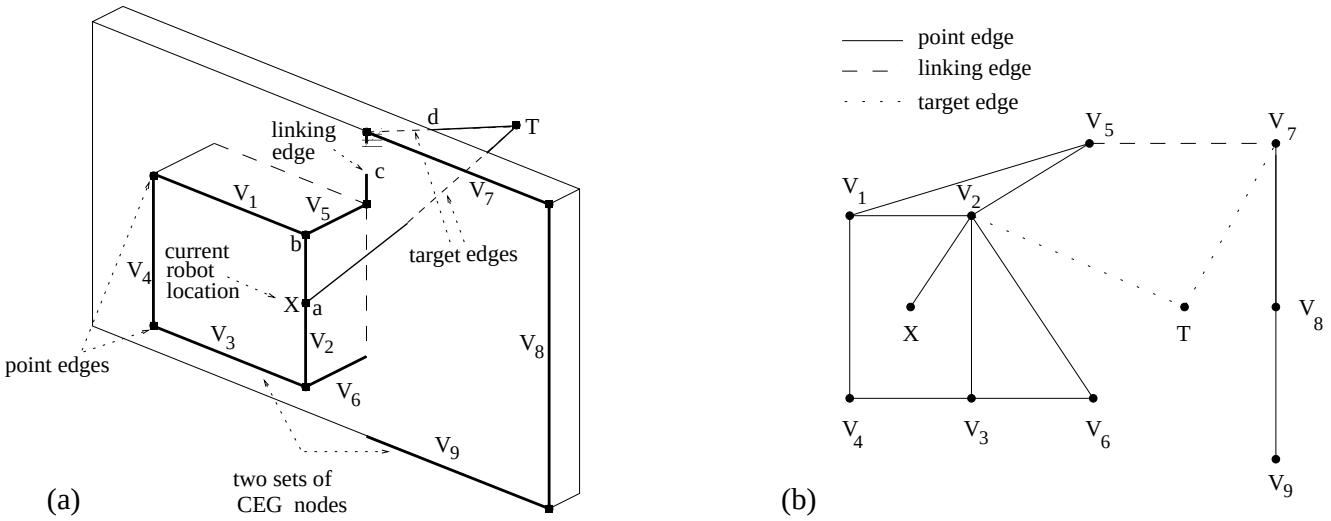


Fig. 13. (a) The CEG components. Two disjoint sets of CEG nodes are visible from the robot location at X, and a linking edge connects them. (b) The corresponding CEG graph. (Only representative target edges are shown.)

separate visible surface patches from invisible surface patches. Much like the CEG, in a polyhedral environment, the visible rim consists of convex obstacle edges and line segments generated by occluding convex obstacle edges. However, the line segments generated by occlusion vary continuously as the robot moves in the environment. In contrast, the CEG consists only of convex obstacle edges that never change their position. The description and maintenance of the visible rim is therefore significantly harder than that of the CEG.

Appendix B. Proof Details

The following lemma characterizes the shortest path in three dimensions.

LEMMA 1. The shortest path in a three-dimensional polyhedral environment is piecewise linear, and the path's vertices lie only on convex obstacle edges.

Proof. Using the standard tool of path-length variation (Thorpe 1979, p. 164), it can be verified that the shortest path in a three-dimensional polyhedral environment is piecewise linear, such that the path's vertices lie on obstacle edges. We now show that the path's vertices must lie on convex obstacle edges. Let $Q = [q_1, \ldots, q_n]$ be the vertices along the shortest path between two points q_1 and q_n. Assume by contradiction that some vertex q_i of Q lies on a concave edge, where $2 \leq i \leq n-1$. Consider the two-dimensional plane A defined by the points q_{i-1}, q_i and q_{i+1}. The three-dimensional polyhedral obstacles in the environment induce two-dimensional polygonal obstacles in A. In particular, the point q_i becomes a concave obstacle vertex in A. The optimality of Q in three dimensions implies that the path segment $[q_{i-1}, q_i, q_{i+1}]$ must be the shortest path in A from q_{i-1} to q_{i+1}. But the path $[q_{i-1}, q_i, q_{i+1}]$ is not optimal in A, since the locally optimal path in the plane passes only through convex obstacle vertices (Liu and Arimoto 1992). Hence, Q is not the shortest path—a contradiction. □

The following lemma associates with every switch point P_i a local minimum of $d(w, T)$.

LEMMA 2. Every switch point P_i, where the robot switches from motion toward the target to obstacle surface traversal, has a corresponding unique point M_i, which is a local minimum of $d(w, T)$, such that $d(M_i, T) \leq d(P_i, T)$.

Proof. When the robot switches to obstacle surface traversal at P_i, there must be a blocking obstacle between P_i and T. Otherwise, the robot can reach T directly from P_i, and no switching to obstacle surface traversal would occur. Let v_{cross} denote the point where the line segment $[P_i, T]$ crosses the blocking surface (Fig. 14). By construction, $d(v_{cross}, T) \leq d(P_i, T)$. The motion toward the target is terminated because the feasible subcontour has become empty. Thus, $d(P_i, T) < d(y, T)$ holds for every point y on the blocking contour. But $d(v_{cross}, T) \leq d(P_i, T)$. Hence, $d(v_{cross}, T) < d(y, T)$ for every point y on the blocking contour. The function $d(w, T)$ is continuous, and the blocking surface is a connected compact set (i.e., closed and bounded) whose boundary is precisely the blocking contour. Since every continuous function attains a minimum on a compact set, there exists a point where $d(w, T)$ attains its minimum on the blocking surface. Furthermore, the local-minimum point, M_i, satisfies $d(M_i, T) \leq d(v_{cross}, T) \leq d(P_i, T)$. Finally, if there are several local minima, the one that is closest to T is chosen, so that a unique local-minimum point is associated with P_i. □

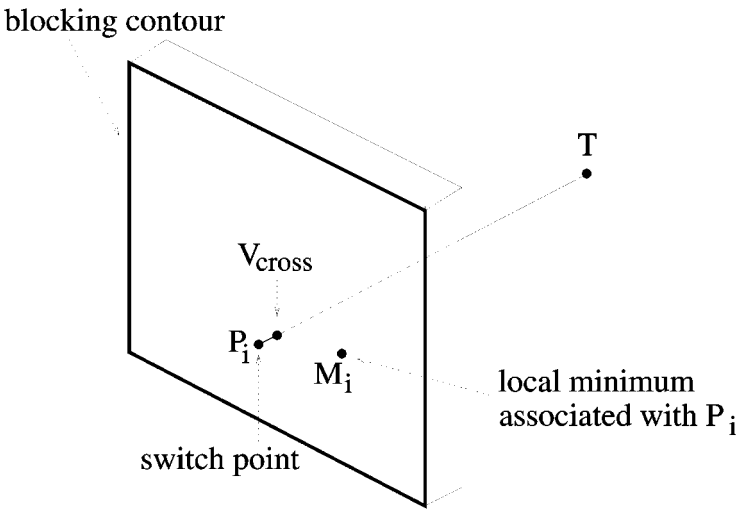


Fig. 14. The feasible subcontour becomes empty at the switch point P_i, and $d(w, T)$ has a local minimum at M_i on the blocking surface.

The final lemma asserts that the distance to the target decreases monotonically between successive local-minimum points.

LEMMA 3. The distance to the target decreases monotonically between successive local-minimum points, that is, $d(M_{i+1}, T) < d(M_i, T)$.

Proof. According to Lemma 4, each switch point P_i is associated with a unique local-minimum point M_i. While moving from P_i to P_{i+1}, the robot first traverses an obstacle surface, then leaves the obstacle and performs a transition phase. The transition phase is followed by motion toward the target, which ends at the switch point P_{i+1}. By construction, the entire blocking surface is visible from P_i. Since M_i lies on the blocking surface, it is visible from P_i. Hence, the minimum distance to the target observed by the robot satisfies $d(p_{min}, T) \leq d(M_i, T)$. The surface traversal is terminated when the exit condition, $d(V_{exit}, T) < d(p_{min}, T)$, holds true. The robot then performs the transition phase and moves toward V_{exit} until it reaches a point Z_i, which satisfies $d(Z_i, T) < d(p_{min}, T)$. Thus, $d(Z_i, T) < d(M_i, T)$ at the end of the transition phase. From Z_i, the motion toward the target is resumed until the switch point P_{i+1} is reached. Since the distance $d(X, T)$ decreases monotonically during motion toward the target, $d(P_{i+1}, T) \leq d(Z_i, T)$, and consequently $d(P_{i+1}, T) < d(M_i, T)$. Based on Lemma 4, $d(M_{i+1}, T) \leq d(P_{i+1}, T)$, and consequently $d(M_{i+1}, T) < d(M_i, T)$. □

References

Blake, A., Zisserman, A., and Cipolla, R. 1992. *Visual Exploration of Free-Space*. Cambridge, MA: MIT Press.

Canny, J. F. 1988. *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press.

Chatila, R. 1995. Deliberation and reactivity in autonomous mobile robots. *Robotics and Autonomous Systems* 16:197–211.

Choi, J., Sellen, J., and Yap, C. K. 1994. Approximate Euclidean shortest path in 3-space. *10th Annual ACM Symposium on Computational Geometry*, pp. 41–48.

Choset, H., and Burdick, J. W. 1995. Sensor based planning, part ii: Incremental construction of the generalized voronoi graph. *IEEE Conference on Robotics and Automation*, Nagoya, Japan, May, pp. 1643–1649.

Connoly, C. 1985. The determination of next best views. *IEEE Conference on Robotics and Automation*, pp. 432–435.

Cox, J., and Yap, C. K. 1988. On-line motion planning: Moving a planar arm by probing an unknown environment. Technical report, Courant Institute of Mathematical sciences, New York University, July.

Crane, C., Duffy, J., and Carnahan, T. 1991. A kinematic analysis of the space station remote manipulator system. *J. Robotics Systems* 8(5):637–658.

Crowley, J. L., and Demazeau, Y. 1993. Principles and techniques for sensor data fusion. *Signal Processing* 32:5–27.

Foux, G., Heymann, M., and Bruckstein, A. 1993. Two dimensional robot navigation among unknown stationary polygonal obstacles. *IEEE Transactions on Robotics and Automation* 9(1):96–102.

Kamon, I. 1997. *Locally Optimal Sensor-Based Robot Navigation*. Ph.D. thesis, Technion—Israel Institute of Technology, Technion City, Israel.

Kamon, I., Rimon, E., and Rivlin, E. Forthcoming. Tangentbug: A range-sensor based navigation algorithm. *International Journal on Robotic Research*.

Kutulakos, K. N., Dyer, C. R., and Lumelsky, V. J. 1994. Provable strategies for vision-guided exploration in three dimensions. *IEEE Conference on Robotics and Automation*, San Diego, CA, May, pp. 1365–1372.

Kutulakos, K. N., Lumelsky, V. J., and Dyer, C. R. 1993. Vision guided exploration: A step toward general motion planning in three dimensions. *IEEE Conference on Robotics and Automation*, Atlanta, GA, May, pp. 289–296.

Laubach, S. L., Burdick, J. W., and Matthies, L. 1998. An autonomous path planner implemented on the rocky7 prototype microrover. *IEEE Conference on Robotics and Automation*, pp. 292–297.

Liu, Y. H., and Arimoto, S. 1992. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *International Journal of Robotic Research* 11(4):376–382.

Lozano-Perez, T., and Wesley, M. A. 1979. An algorithm for planning collision free paths among polyhedral obstacles. *Communications of the ACM* 22(10):560–570.

Lumelsky, V. J. 1991. A comparative study on the path length performance of maze-searching and robot motion planning algorithms. *IEEE Transactions on Robotics and Automation* 7(1):57–66.

Lumelsky, V. J., and Stepanov, A. A. 1987. Path-planning strategies for a point mobile automaton moving amidst obstacles of arbitrary shape. *Algorithmica* 2:403–430.

March, E., and Chaumette, F. 1997. Active sensor placement for complete scene reconstruction and exploration. *IEEE Conference on Robotics and Automation*, pp. 743–750.

Maver, J., and Bajcsy, R. 1993. Occlusions as a guide for planning the next view. *IEEE Transactions on PAMI* 15(5):417–433.

Nayar, S. 1997. Catadioptric omnidirectional cameras. *CVPR97*, pp. 331–339.

Noborio, H., and Yoshioka, T. 1993. An on-line and deadlock-free path-planning algorithm based on world topology. *IEEE/RSJ Conference on Intelligent Robots and Systems, IROS*, pp. 1425–1430.

Papadimitriou, C. H. 1985. An algorithm for shortest path motion in three dimensions. *Information Processing Letters* 20:259–263.

Rao, N.S.V., Iyengar, S. S., Oommen, B. J., and Kashyap, A.R.L. 1988. On terrain model acquisition by a point robot amidst polyhedral obstacles. *IEEE Transactions on Robotics and Automation* 4(4):450–455.

Rieger, J. H. 1990. The geometry of view space of opaque objects bounded by smooth surfaces. *Artificial Intelligence* 44:1–40.

Rimon, E. 1997. Construction of c-space roadmaps from local sensory data. What should the sensors look for? *Algorithmica* 17(4):357–379.

Rohnert, H. 1986. Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters* 23:71–76.

Sankaranarayanan, A., and Vidyasagar, M. 1991. Path planning for moving a point object amidst unknown obstacles in a plane: The universal lower bound on worst case path lengths and a classification of algorithms. *IEEE Conference on Robotics and Automation*, Sacramento, CA, April, pp. 1734–1941.

Stentz, A. 1994. Optimal and efficient path planning for partially known environments. *IEEE Conference on Robotics and Automation*, San Diego, CA, May, pp. 3310–3317.

Svoboda, T., Pajdla, T., and Hlavac, V. 1998. Epipolar geometry for panoramic cameras. *ECCV98*, pp. 218–231.

Technion - Israel Institute of Technology. 1997. Irit 7.0 User's Manual. Technion City, Israel, www.cs.technion.ac.il/irit, February.

Thorpe, J. A. 1979. *Elementary Topics in Differential Geometry*. New York: Springer-Verlag.

TITAN. 1996. A telerobotic manipulator for nuclear inspection and maintenance. Schilling Robotic Systems.

Whaite, P., and Ferrie, F. P. 1991. From uncertainty to visual exploration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(10):1038–1049.

Whitney, H. 1955. On singularities of mappings of Euclidean spaces. I. mappings of the plane into a plane. *Annals of Mathematics* 62(3):374–410.