Vision-Guided Exploration: A Step Toward General Motion Planning in Three Dimensions

Kiriakos N. Kutulakos
Charles R. Dyer
Vladimir J. Lumelsky

Technical Report #1111

September 1992

Vision-Guided Exploration: A Step Toward General Motion Planning in Three Dimensions

Kiriakos N. Kutulakos
Charles R. Dyer

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53706

Vladimir J. Lumelsky

Mechanical Engineering Department
University of Wisconsin
Madison, Wisconsin 53706

Technical Report #1111
September 1992

Abstract

We present an approach for solving the path planning problem for a mobile robot operating in an unknown, three dimensional environment containing obstacles of arbitrary shape. The main contributions of this paper are (1) an analysis of the type of sensing information that is necessary and sufficient for solving the path planning problem in such environments, and (2) the development of a framework for designing a provably-correct algorithm to solve this problem. No such analysis is currently available for the case where the robot is able to freely move in space (i.e., has three degrees of freedom in position). Working from first principles, without any assumptions about the environment of the robot or its sensing capabilities, our analysis shows that the ability to explore the obstacle surfaces (i.e., to make all their points visible) is intrinsically linked with the ability to plan the motion of the robot. We argue that current approaches to the path planning problem with incomplete information simply do not extend to the general three-dimensional case, and that qualitatively different algorithms are needed. We also present a simple algorithm for solving the three-dimensional path planning problem, under the assumption that an exploration algorithm is available to the robot.

The support of the National Science Foundation under grants IRI-9022608 and IRI-8805943 is gratefully acknowledged.

1 Introduction

The goal of our work is the development of strategies for real-time, purposeful, robust, and provably-correct automatic motion planning in unknown environments where three-dimensional reasoning and visual sensing is necessary. This type of reasoning is necessary when a mobile robot must control its position for the purpose of reaching a desired location, learning the shape of an unknown object, producing a map of a three-dimensional environment, or simply surveying it.

In this paper we consider the *path planning* or *find-path* problem, a problem fundamental to the task of moving within a complex environment. We assume that the environment is three-dimensional space containing obstacles that are finite volumes bounded by closed surfaces of arbitrary shape. We assume that the robot is a point and that it is equipped with one or more sensors (e.g., a camera or a range sensor). Its goal is to plan a collision-free path from an initial to a target location if such a path is possible, or to report that such a path does not exist.

A crucial issue in motion planning is the type of information the robot has or is able to recover about its environment through its sensors. Motivated by early artificial intelligence approaches to problem-solving and planning, a large body of research in robotics considered approaches following what could be called an *act-after-thinking* strategy. These approaches emphasized the mutual independence between sensing and action and generated interest in solutions to motion planning problems where complete information is available at the time of robot action or decision making. The Piano Movers problem [1] was subsequently formulated in order to solve the path planning problem in cases where the environment of the robot was already accurately known.

The difficulties involved in recovering complete information about the environment *a priori* (e.g., about unstructured or cluttered terrains, or undersea and space environments) motivated an alternative approach to robotic motion planning. Its underlying principle is that intelligent behavior is the result of a collection of simple reactions to a complex world [2]. This approach follows a *purposive* [3], *act-while-thinking* strategy [4]: Instead of trying to achieve the general recovery goal, attack the motion planning problem directly by (1) using only the sensor information necessary for planning the motion of the robot, and (2) considering robotic motion planning as a continuous process where sensing and action are tightly coupled. The approach advocates the need for real-time robot control with incomplete information [5] or under uncertainty [6], and real-time spatiotemporal processing of the sensor input [7,8].

In this paper we study the path planning problem under the purposive, act-while-thinking paradigm. The main contributions of this paper are (1) an analysis of the type of sensing information that is *necessary and sufficient* for solving the path planning problem in an unknown three-dimensional environment, and (2) the development of a framework for solving the path planning problem in such environments. No such analysis is currently available for the case where the mobile robot is able to freely move in space (i.e., has three degrees of freedom in position).

Although the analysis we present is theoretical, its results have important practical consequences. A major consequence is that assumptions about the capabilities of the robot and its sensors currently used in path-planning algorithms for two-dimensional environments simply do not extend to the three-dimensional case. Furthermore, our analysis indicates that visual sensing[1] is far more important for planning the motion of the robot in unknown and

[1]We use the term "visual sensing" in this context to characterize sensing mechanisms that allow us to recover shape information about the portions of the obstacle surfaces that are visible to the robot from its current position. These portions contain all points for which the open line segment connecting them to the robot does not intersect any obstacle. In our discussion "visual sensors" include cameras, where only information about the *projected* shape of the visible surfaces is directly available, as well as more powerful mechanisms such as range sensors, that directly provide distance information for visible surface points within a finite ball around the robot.

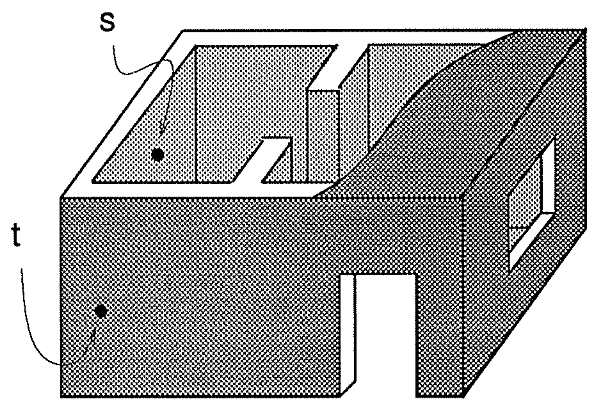

Figure 1: Motion planning on the surface of a building. The start position **s** is on the interior surface of the building while the target position **t** is on the exterior surface. The interior and exterior surfaces of the building are connected through the door and the window.

unstructured three-dimensional environments than previously thought (e.g., it provides the robot with more powerful motion planning capabilities than tactile sensing). In particular, we show that in order for the robot to be able to plan its motion in such environments it must be able to visually explore them. This stresses the importance of solving the *visual exploration problem*, which is the problem of making all points in the environment visible by planning a finite-length path for the robot. In addition, these facts point to three-dimensional path-planning algorithms that are qualitatively different from the algorithms currently employed in robotics and that contain visual information processing as an important and indispensable component.

Our emphasis is on developing deterministic strategies for solving the path planning problem that are not heuristic-based, but rather possess predictable properties (e.g., correctness and bounded length of generated paths) even in geometrically-complex three-dimensional environments. The major problems that arise while trying to extend current approaches to the cases we are considering are outlined below.

First, current approaches consider path planning as a problem independent of the visual exploration problem. An important consequence of our analysis in Section 2, which constitutes the main part of this paper, is that in order to plan a finite-length path to an arbitrary location in the environment or to determine whether that location is unreachable, the robot must in general be able to visually explore the environment. An intuitive explanation of this result can be given by considering the path planning problem in Figure 1. If the robot is an "ant" constrained to move on the wall of the building, it must find either the window or the door before it is able to reach the target position $\mathbf{t}$. To do this it may need to explore the entire interior surface of the building. This example is a manifestation of a deeper result that follows from our analysis which states that the path planning problem is unsolvable, in general, for a robot moving on an unknown surface for which the only information available to the robot is the robot's current three-dimensional coordinates and the coordinates of the target location. Existing algorithms solve the path planning problem based solely on this information only for the cases where the robot moves on a plane or on a surface for which some geometrical information is available (e.g., that it is a torus [9]), or when known relevant landmarks (e.g., a door) are available to guide the robot's motion [8, 10]. However, the building example above illustrates that the path planning problem needs an approach very different from current approaches when no shape information about the surface on which the robot moves is available (e.g., a sunken ship or the ocean floor) and no landmarks are present, requiring an exploratory process. This is also the case for a robot moving freely in three-dimensional space cluttered with obstacles of arbitrary shape.

The results of our analysis in Section 2 establish the importance of addressing the visual exploration problem in arbitrary three-dimensional environments, since solving this problem is a fundamental requirement for solving the path planning problem. This leads us to the analysis of the visual exploration problem. Even though the visual exploration problem has been considered previously for the case where the robot moves in a three-dimensional space cluttered with obstacles, only polyhedral obstacles have been treated. The algorithms rely on the fact that the obstacles consist of a finite collection of planar faces, and produce paths whose lengths diverge in the limit. For example, the exploration algorithm proposed by Rao [11] has a complexity depending on the number of faces in the environment's description. This description, however, becomes infinite when the obstacles are bounded by smooth surfaces. Consequently, exploration becomes an infinitely complex task even when the environment contains geometrically simple surfaces (e.g., a sphere). On the other hand, previous work in computer vision that addresses the visual exploration problem (e.g., [12]) has not formally studied this problem and has not considered issues of correctness or convergence in arbitrary environments.

The remainder of this paper is organized as follows. The next section presents the main result of the paper, stating that the three-dimensional path planning problem requires in general an exploratory process. The implications of this result are then discussed in Section 2.1. Motivated by these results and by the limitations of current approaches to the path planning problem, we consider this problem in conjunction with the visual exploration problem. To this end, Section 3 presents a simple algorithm for solving the path planning problem in arbitrary three-dimensional environments under the assumption that an exploration algorithm is available to the robot.

2 Unsolvability of Non-Exploratory Path Planning in $\Re^3$

In this section we show that, in general, path planning in three-dimensional environments requires an exploratory process. We formally develop the notion of an "exploratory" algorithm in an abstract setting, using an abstract definition of the sensing mechanism of the robot.

Without loss of generality we assume that the environment contains a single connected surface S. Furthermore, we assume that S is closed, has an arbitrary shape, and bounds an open, finite and connected volume V_S.[2] We define an automaton α able to plan its motion in the space $\Re^3 - V_S$ as follows:

Definition 2.1 (Sensing mechanism) The *sensing mechanism* of α is described by a function $\mathfrak{S}$ assigning to each point x in $\Re^3 - V_S$ a subset $\mathfrak{S}(x)$ of $\Re^3 - V_S$, such that when α is positioned at x it can determine the three-dimensional coordinates of all points in $\mathfrak{S}(x)$.

Definition 2.2 (Path of the automaton) The *path* p traced by the automaton is a continuous curve in $\Re^3 - V_S$ of finite length. The point $\mathbf{s} = p(0)$ is the start position of α. The last point on p corresponds to the current position $\mathbf{c}$ of the automaton.

Definition 2.3 (Memory of the automaton) Given a path p of the automaton, its *memory* $\mathcal{M}(p)$ is described by a subset of $\cup_{x \in p - \{\mathbf{c}\}} \mathfrak{S}(x)$, such that when α traces path p it can store the three-dimensional coordinates of all points in $\mathcal{M}(p)$.

The definition of the automaton's sensing function expresses the automaton's ability to determine the three-dimensional coordinates for all points belonging to $\mathfrak{S}(\mathbf{c})$. The idea behind the definition of the automaton's memory is that the automaton is capable of storing the coordinates of some (or all) of the points it sensed from along its path.

[2] "Hollow" objects are not allowed under this definition, since this would imply that the surface bounding the object is not connected.

Example 2.1 (Tactile sensing) If $\mathfrak{S}(\mathbf{c}) = \{\mathbf{c}\}$ for $\mathbf{c} \in \Re^3 - V_S$, the automaton can only determine the three-dimensional coordinates of its current position, an assumption used in the path-planning literature (e.g., see [5]).

Example 2.2 (Range sensing) If $\mathfrak{S}(\mathbf{c}) = \{x \in \Re^3 | x \in B(\mathbf{c}, r) \text{ and } x \text{ is visible from } \mathbf{c}\}$, the sensing mechanism of the automaton corresponds to a range sensor that can determine the coordinates of all visible points within a ball of radius r. Recall that a point x is considered visible if the open line segment connecting it to $\mathbf{c}$ does not intersect V_S.

Example 2.3 If $\mathfrak{S}(\mathbf{s}) = \Re^3 - V_S$ then the automaton has complete information about the environment before it starts planning its path.

The above examples show that the definition of the sensing function $\mathfrak{S}$ of the automaton is very general (e.g., it does not restrict the sensing ability of the robot to purely "local" sensing). This generality is intentional since our purpose is to identify those properties of the sensing function that are crucial for determining the automaton's ability to solve the path planning problem. We are interested in sensing mechanisms that are less powerful than that of Example 2.3.

The automaton can now be formally defined as follows:

Definition 2.4 (Automaton) An automaton α is a 5-tuple $(\mathfrak{S}, \mathcal{M}, \mathcal{A}, \mathcal{I}, p)$ where

$\mathfrak{S}$	is the sensing mechanism of the automaton
$\mathcal{M}$	is the memory of the automaton
$\mathcal{A}$	is the deterministic algorithm generating a new position for the automaton
$\mathcal{I}$	is the input to algorithm $\mathcal{A}$ (defined below)
p	is the path in $\Re^3$ traced by the automaton.

In order to define the path planning problem we also need to define the notion of point reachability:

Definition 2.5 (Reachable points) Two points $x_1, x_2 \in \Re^3$ are reachable from each other iff they can be connected by a continuous curve that does not intersect V_S. Equivalently, x_1, x_2 are reachable iff they both belong to $\Re^3 - V_S$.

An automaton that solves the path planning problem is an automaton such that (1) when its current position is $\mathbf{c}$, algorithm $\mathcal{A}$ accepts as input $\mathcal{I}$ the tuple $(\mathbf{s}, \mathbf{t}, \mathfrak{S}(\mathbf{c}), \mathcal{M}(p))$, where p is the path already traced by the automaton, $\mathbf{s} \in \Re^3 - V_S$ is the initial position of the automaton, and $\mathbf{t}$ is the target position, (2) if $\mathbf{t}$ is reachable from $\mathbf{s}$, then a path $p(\mathbf{s} \rightarrow \mathbf{t})$ connecting $\mathbf{s}$ to $\mathbf{t}$ can be generated by $\mathcal{A}$, and (3) if $\mathbf{t}$ is not reachable from $\mathbf{s}$, then $\mathcal{A}$ terminates after it has generated a finite-length path.

We now focus on a property of algorithm $\mathcal{A}$ that is crucial to our analysis below.

Definition 2.6 (Exploratory algorithm) We call $\mathcal{A}$ *exploratory* iff for any surface S we can choose a positive number M_S such that the following two conditions are satisfied:

1. For any path p generated by $\mathcal{A}$, if length$(p) > M_S$ then for all open sets $U \subset S$,

$$U \cap \left[\bigcup_{x \in p} \mathfrak{S}(x) \right] \neq \emptyset$$

2. At least one path of length greater than M_S can be generated by $\mathcal{A}$.

We call $\mathcal{A}$ *non-exploratory* if at least one of these two conditions is not satisfied.

This definition of an exploratory algorithm states that the automaton is capable of sensing all points on the surface (except for pathological sets) if a sufficiently large, but finite, path is generated. Note that the exploratory property imposes very strict constraints on the algorithm $\mathcal{A}$: It does not simply require $\mathcal{A}$ to be capable of generating a path that allows the automaton to sense all points on the surface; it requires that *all* paths generated by $\mathcal{A}$ having length greater than M_S have this property. Intuitively, this means that the algorithm must control the motion of the automaton by always taking into account the points on the obstacle surfaces that have already been sensed. Also note that the definition does not depend on the characteristics of the path generated (e.g., whether it is a path between a starting position s of the automaton and an *a priori* determined target position $\mathbf{t}$).

The main goal of this section is to prove the following theorem:

Theorem 2.1 *Let* $\alpha = (\mathfrak{S}, \mathcal{M}, \mathcal{A}, \mathcal{I}, p)$ *be an automaton, where*

1. $\mathcal{A}$ *is a deterministic algorithm*
2. $\mathfrak{S}(\mathbf{c})$ *contains* $\mathbf{c}$ *for any* $\mathbf{c} \in \Re^3 - V_S$
3. $\mathcal{M}(p) = \bigcup_{x \in p} \mathfrak{S}(x)$
4. $\mathcal{I} = (\mathbf{s}, \mathbf{t}, \mathfrak{S}(\mathbf{c}), \mathcal{M}(p))$
5. $\mathcal{A}$ *can generate a path between* $\mathbf{c}$ *and any point in* $\mathfrak{S}(\mathbf{c})$
6. $\mathcal{M}(p)$ *is closed for any finite-length path* p.

Then α *solves the path planning problem iff* $\mathcal{A}$ *is exploratory.*

Condition 2 states that the automaton has available the three-dimensional coordinates of its current position. Conditions 3 and 5 allow the automaton to store the coordinates of all points it already sensed along its path and to plan a path to any of those points. Condition 6 ensures that the robot can determine the boundary of the points already sensed; together with Definition 2.6 it ensures that an exploratory algorithm will allow the automaton to sense all points on the obstacle surface.

We prove the 'only if' part of Theorem 2.1, since the other part is immediate by Conditions 3 and 5. We prove this by contradiction, assuming that there is a non-exploratory algorithm that allows α to solve the path planning problem. In particular, we prove the following proposition:

Proposition 2.1 *Let $\alpha = (\mathfrak{S}, \mathcal{M}, \mathcal{A}, \mathcal{I}, p)$ be an automaton satisfying the conditions of Theorem 2.1. If $\mathcal{A}$ is non-exploratory,*

1. *There exists a surface $\hat{S}$, and $\mathbf{s} \in \Re^3 - V_{\hat{S}}$ such that the length of the path generated by $\mathcal{A}$ for any point $\mathbf{t} \in V_{\hat{S}}$ is unbounded.*
2. *Given $M > 0$, there is a point $\mathbf{t}$ on $\hat{S}$ such that $\mathcal{A}$ plans a path between $\mathbf{s}$ and $\mathbf{t}$ of length greater than M.*

This proposition implies that determining whether or not $\mathbf{t}$ is reachable is an unsolvable (or undecidable [13]) problem since if $\mathbf{t}$ is not reachable, the automaton's algorithm will not terminate. Refer to the Appendix A for a proof of the proposition.

2.1 Implications of Theorem 2.1

The intuitive result that follows from Theorem 2.1 is that the path planning problem in arbitrary three-dimensional environments is in general unsolvable if the robot does not possess an ability to explore the surface of an obstacle. An important consequence of this result is that apart from the algorithmic machinery that must be available to the robot, certain constraints are put on the sensing mechanisms that the robot needs to plan its path.

An important constraint imposed by the theorem is that the robot must be able to sense all points on the surface (i.e., a two-dimensional set of points) from a one-dimensional set of positions corresponding to the robot's path. This implies that the sensing mechanism of the robot must fulfill either (or both) of the conditions below for an arbitrary surface:

- The robot can sense a two-dimensional set of points on the surface from a discrete number of positions along its path.
- From each position in a one-dimensional subset of its path, the robot can sense a one-dimensional set of points on the surface it did not previously sense.

The immediate consequence of these conditions is that tactile sensing, whereby the robot can determine the coordinates of a point on the obstacle surface by means of single-point contact, is not sufficient to guarantee the successful exploration of an arbitrary surface. Hence, tactile sensing cannot guarantee the correctness of a three-dimensional path planning algorithm; more powerful sensing mechanisms are necessary to deal with the path planning problem in three dimensions. On the other hand, the above conditions point directly to algorithms that use visual sensing (i.e., either a camera or a range sensor) for guiding the exploration and the path-planning processes: The first condition leads to strategies where the path planning process is guided by observing the obstacle surfaces from a discrete set of positions and planning the motion of the robot between them (e.g., similar to [12]). The second condition leads to the development of strategies that are based on the detection of one-dimensional curves that can "slide" on the surface as the robot moves (e.g., the occlusion boundary [7]).

The above conditions also point to a two-dimensional path planning problem that has been hitherto not considered: The problem of planning the motion of the robot between two points on an unknown surface. Although the robot in this case moves in a two-dimensional space, the path planning problem cannot be solved using traditional path planning algorithms that assume the robot moves on a plane or a *known* surface (e.g., a torus). In fact, the above conditions show that this problem is unsolvable even if the robot is equipped with visual sensors, because the robot will not be able to explore the surface by tracing a finite length path. For example, if the robot moves on a convex part of the surface, only the point of contact of the surface with the robot can be sensed, which is a zero-dimensional set. An intuitive explanation of this result is that the problem is an overconstrained version of the three-dimensional path planning problem, where the robot must reach a point described by its three-dimensional coordinates while being constrained to move on a fixed (but unknown) surface.This implies that the robot must in general move above the surface, planning its motion in three-dimensional space, in order to reach the target position.

An important point in Theorem 2.1 is that no assumptions are being made about the nature of the space in which the path planning problem is formulated. For example, the space in which path planning is performed does not have to be the workspace of the robot but could instead be a configuration space. This amplifies the importance of our result since it implies that some motion planning problems that are cast as path planning problems in three-dimensional configuration spaces may require the ability to explore the surface of the configuration space obstacles. Non-exploratory path planning algorithms in three-dimensional configuration spaces have been developed that take into account the natural constraints on the shape of the configuration space obstacles imposed by the geometry of the robotic manipulator [9,14]. However, there are problems where the configuration space obstacles can have arbitrary shapes. For example, the problem of planning the path of a box between obstacles in a plane can be transformed to the problem of planning the path of a point robot in a three-dimensional configuration space containing arbitrarily-shaped obstacles. An important direction for future research will be to design exploratory algorithms for this and other similar configuration space path planning problems.

3 Path Planning in $\Re^3$

Having identified the key characteristics of algorithms that can solve the path planning problem in three dimensions, we now consider the issue of how to design such an algorithm. We assume that the environment of the robot is $\Re^3$, containing a number of obstacles that are finite volumes bounded by closed surfaces of arbitrary shape. Obstacles in the environment do not touch each other. Furthermore, we assume that (1) any ball of finite radius intersects only a finite number of such obstacles, and (2) the intersection of any plane with an obstacle contains only a finite number of connected regions.

We will assume that the robot is modeled as a point automaton α. The automaton has available an exploratory algorithm $\mathcal{A}_E$, discussed in Section 4, for exploring the surface of any obstacle by tracing a finite-length path. The goal of this section is to design a provably-correct exploratory algorithm $\mathcal{A}_{3D}$ for α that will use $\mathcal{A}_E$ as a component to solve the three-dimensional path planning problem. Here we focus on the case where the sensing mechanism of the robot is a range sensor, i.e., the robot is able to determine the three-dimensional coordinates of all points on the obstacle surfaces such that (1) they lie within a ball of fixed radius, and (2) the open line connecting them to the position of the robot does not intersect any obstacle.

The main idea of our approach is to decompose the path planning problem in three dimensions into two independent subproblems:

- A planar path planning problem, solved by an algorithm $\mathcal{A}_{2D}$, where the robot must either plan a path between two points $\mathbf{s}$ and $\mathbf{t}$ in a given plane by tracing a finite-length path, or determine that the two points are not reachable from each other.

- A three-dimensional exploration problem, solved by algorithm $\mathcal{A}_E$, where the robot explores an obstacle surface until either the robot's position or a surface point sensed by the robot satisfies a given condition.

The remainder of this section describes how algorithms $\mathcal{A}_{2D}$ and $\mathcal{A}_E$ for solving these two subproblems can be combined to produce a path planning algorithm for the three-dimensional problem.

3.1 Decomposition of the Path Planning Problem in $\Re^3$

Let $\mathbf{s}$ and $\mathbf{t}$ be the start and target position of the robot and let x be a point in $\Re^3$ such that $\mathbf{s}, \mathbf{t}$ and x are not collinear. These three points define a plane Π in $\Re^3$ that possibly intersects the obstacles in the environment. Although the target location $\mathbf{t}$ may be reachable from $\mathbf{s}$, it may or may not be reachable by a collision-free path on Π. If $\mathbf{s}$ and $\mathbf{t}$ are reachable on Π, one of the known planar path planning algorithms (e.g., [5]) will be able to solve this path planning problem. On the other hand, if $\mathbf{t}$ is not reachable, the robot must plan its motion by moving outside of Π (Figure 2). The idea behind the decomposition of the three-dimensional path-planning problem is to consider it as a collection of two-dimensional path-planning problems in Π and a series of exploratory motions that force the robot to move outside of Π.

Definition 3.1 (Π-reachable) Let Π be a plane in $\Re^3$. Points $\mathbf{s}$ and $\mathbf{t}$ are called Π-*reachable* if they can be connected by a curve in Π that does not pierce any obstacle.[3]

In general, plane Π will intersect a number of obstacles in the environment. The intersection of Π with an obstacle $\mathcal{O}$ will be a collection of closed curves (not necessarily simple), lines, points, or even two-dimensional regions of Π (e.g., when Π touches the face of a cube). Since the robot is allowed to move on the surface, areas of contact of the surfaces with Π that

[3]We will assume that the robot can move on the obstacle surfaces, and therefore paths that are tangential to the obstacle surfaces or lie on these surfaces will be considered acceptable.

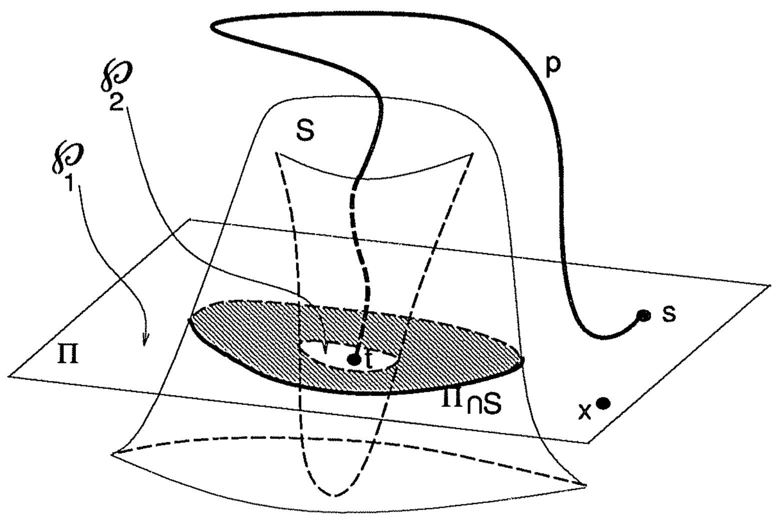

Figure 2: Path planning around a crater-like surface. Points **s** and **t** are reachable but not Π-reachable. The robot must plan a path outside plane Π in order to reach t (e.g., path p). The two curves in $\Pi \cap S$ are the only obstacles presented to a robot constrained to move in Π. These curves define two equivalence classes $\wp_1, \wp_2$ of Π-reachable points.

are isolated lines, points or two-dimensional regions are not considered obstacles. Therefore, the obstacles for a robot constrained to move in Π will only consist of closed curves. Π-reachability is defined in terms of these obstacles. It partitions points in Π not belonging to obstacle interiors into equivalence classes, where each equivalence class contains points that are Π-reachable from each other (Figure 2).

Definition 3.2 (Reachability region) The set $\{x : x$ and $\mathbf{t}$ are Π-reachable$\}$ is called the *reachability region* of **t** in Π.

Clearly, if the robot can reach from s any point within the reachability region of **t** in Π, then any planar path planning algorithm that is consistent with the definition of the obstacles in Π (i.e., non-polyhedral and arbitrarily-shaped closed curves) can complete the path from s to **t**. We will use the exploration algorithm $\mathcal{A}_E$ to reach the reachability region of **t** in Π. This region is bounded by curves in the intersection of Π with the obstacles in the environment. Therefore, if the robot is able to sense points belonging to any of these curves using $\mathcal{A}_E$, the robot will be able to reach a position on Π that belongs to the reachability region of $\mathbf{t}$.

3.2 Algorithm $\mathcal{A}_{2D}$

Simply for the sake of specificity we assume that the robot uses algorithm Bug1 [5] for solving the two-dimensional motion planning subproblem. This algorithm assumes the robot can determine its three-dimensional coordinates and knows the coordinates of the target location. The robot can also determine when it is in contact with an obstacle, and it can move (1) in a straight line toward the target until it contacts an obstacle surface or until it reaches the target, or (2) move along an obstacle boundary in plane Π.

Briefly, a robot using the Bug1 algorithm moves toward the target in a straight line in Π until an obstacle is encountered or until the target is reached. If an obstacle is encountered, the robot moves along the obstacle boundary in Π until it is completely circumnavigated. It then performs a Π-reachability test and if the test is positive it defines a *leave point* on the obstacle boundary and continues moving toward the target in a straight line. If the target is not Π-reachable, the algorithm can determine which obstacle boundary in Π contains the target in its interior.

3.3 Algorithm $\mathcal{A}_E$

We will assume that the robot uses a procedure $EXPLORE(l, d)$ to explore the obstacle surface containing the closed curve $l \in S \cap \Pi$ until a point on S is sensed by the robot such that its distance to $\mathbf{t}$ is less than d.

3.4 Algorithm $\mathcal{A}_{3D}$

We now have a simple algorithm for planning the path of the robot in arbitrary three-dimensional environments:

1. Use algorithm $\mathcal{A}_{2D}$ until either $\mathbf{t}$ is reached or until $\mathcal{A}_{2D}$ determines that $\mathbf{t}$ is not Π-reachable (i.e., $\mathbf{t}$ is contained in the open region of Π bounded by $l \in S \cap \Pi$). If $\mathbf{t}$ is reached, stop. Otherwise, let d be the distance of $\mathbf{t}$ from l.

2. Use algorithm $EXPLORE(l, d)$ to explore the surface of the obstacle containing l until a point $x \in \Pi$ is sensed with distance to $\mathbf{t}$ less than d. If no such point is sensed, stop ($\mathbf{t}$ is unreachable).

3. If a point x is sensed, move on a straight line to x, and continue at Step 1.

We provide a proof of the following lemma in the Appendix, which shows that algorithm $\mathcal{A}_{3D}$ is correct.

Lemma 3.1 *Steps 1-3 will be executed a finite number of times. Furthermore, if* $\mathbf{t}$ *is reachable, the position of the robot when the algorithm terminated will be* $\mathbf{t}$.

4 Exploring Smooth Surfaces in $\Re^3$

The goal of the exploration algorithm $\mathcal{A}_E$ is to generate a path for the robot that allows it to sense, through its visual sensors, an obstacle surface point that satisfies a specific distance constraint from the target location. The key phenomenon limiting the robot's ability to sense such a point from an arbitrary position is that of *occlusion.* Occlusion occurs in environments containing opaque obstacles, where the light reflected from their surfaces cannot go through the obstacles. Consequently, for any position of the robot, some portions of the obstacle surfaces will be *occluded* from view. In general, in order for the robot to induce the visibility of an obstacle surface point satisfying the above distance constraint, the robot must have the ability to visually explore an arbitrary obstacle surface, i.e., to induce the visibility of all surface points that are occluded by generating an appropriate finite-length path for the robot.

Visual exploration of an obstacle surface requires the ability to perform two tasks: (1) to plan the motion of the robot so that it induces the visibility of previously-occluded points on the surface, and (2) to then determine that the exploration goal has been achieved. In order for the robot to perform these two tasks it must represent in some way the set of surface points becoming visible during the exploration process or, alternatively, the set of surface points that remain occluded.

The approach we take allows the robot to explore the environment by following a continuous, collision-free path while continuously monitoring the environment using a camera. The central concept in our approach is the *exploration frontier.* This is the one-dimensional set of points on the surface that bounds the points seen so far during the exploration process. We address the exploration problem by considering the dynamic evolution of the exploration frontier under continuous motion of the robot. Its evolution depends in a simple way on the instantaneous motion of the robot and the dynamics of the visual mapping.

As the robot continuously moves in the environment, the exploration frontier "slides" on the surface, tracing strips. The main idea of the exploration strategy is to have the robot move on a family of parallel planes (e.g., parallel to the plane Π used by algorithm $\mathcal{A}_{3D}$) in order to completely cover the surface with partially overlapping cylindrical strips. This strategy involves two phases. The first phase is a planar motion phase in which the robot explores a cylindrical strip on the surface. In the second phase the robot first decides whether it should move above or below the current motion plane and then moves in the chosen direction to a different plane in the family in order to explore a new strip on the surface. Results from the differential topology of surfaces allow us to represent the surface as a graph and to reduce the exploration process to an abstract graph exploration problem. In this respect our approach is close in spirit to approaches that address the motion planning problem in planar environments by first reducing it to a graph-theoretical problem [4,11]. We use a graph representation of the surface that is a direct generalization of a tree-like representation developed by Koenderink for the qualitative description of gray-scale images [15]. More details on the algorithm and the analysis of its correctness and convergence properties will be presented in a future paper.

5 Concluding Remarks

This paper reveals a crucial link between the path planning problem and the problem of visually exploring a three-dimensional environment. By showing that visual exploration is necessary for successfully planning the motion of a robot in unknown three-dimensional environments, our analysis stresses the importance of integrating exploratory visual sensing with motion planning. This work makes explicit the need for solving the visual exploration problem. Important future research issues will be to study the implications of our analysis for path planning problems in three-dimensional configuration spaces, and to study the visual exploration problem.

References

[1] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. Twentieth Symposium on Foundations of Computer Science*, pp. 421–427, 1979.

[2] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robotics Automat.*, vol. 2, no. 1, pp. 14–23, 1986.

[3] J. Aloimonos, "Purposive and qualitative active vision," in *Proc. Int. Conf. on Pattern Recognition*, pp. 346–360, 1990.

[4] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun, "Dynamic path planning in sensor-based terrain acquisition," *IEEE Trans. Robotics Automat.*, vol. 6, no. 4, pp. 462–472, 1990.

[5] V. J. Lumelsky, "A comparative study on the path length performance of maze-searching and robot motion planning algorithms," *IEEE Trans. Robotics Automat.*, vol. 7, no. 1, pp. 57–66, 1991.

[6] R. A. Brooks, "Visual map making for a mobile robot," in *Proc. IEEE Robotics Automat. Conf.*, pp. 824–829, 1985.

[7] A. Blake, M. Brady, R. Cipolla, Z. Xie, and A. Zisserman, "Visual navigation around curved obstacles," in *Proc. IEEE Robotics Automat. Conf.*, pp. 2490–2495, 1991.

[8] E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine Vision and Applications*, vol. 1, pp. 223–240, 1988.

[9] V. Lumelsky and K. Sun, "A unified methodology for motion planning with uncertainty for 2d and 3d two-link robot arm manipulators," *Int. J. Robotics Research*, vol. 9, no. 5, pp. 89–104, 1990.

[10] P. Stelmaszyk, H. Ishiguro, and S. Tsuji, "Mobile robot navigation by an active control of the vision system," in *Proc. Int. Joint Conf. on Artificial Intelligence*, pp. 1241–1246, 1991.

[11] N. S. Rao, S. S. Iyengar, B. J. Oommen, and R. Kashyap, "Terrain acquisition by point robot amidst polyhedral obstacles," in *Proc. Third Conf. on Artificial Intelligence Applications*, pp. 170–175, 1987.

[12] C. I. Connoly, "The determination of next best views," in *Proc. IEEE Robotics Automat. Conf.*, pp. 432–435, 1985.

[13] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation.* Prentice-Hall, 1981.

[14] K. Sun and V. J. Lumelsky, "Motion planning for three-link robot arm manipulators operating in an unknown three-dimensional environment," in *Proc. Conf. on Decision and Control*, pp. 1019–1026, 1991.

[15] J. J. Koenderink and A. J. van Doorn, "A description of the structure of visual images in terms of an ordered hierarchy of light and dark blobs," in *Proc. Second Int. Visual Psychophysics and Medical Imaging Conf.*, pp. 173–176, 1981.

[16] M. A. Armstrong, *Basic Topology.* Springer-Verlag New York Inc., 1983.

Appendix

A Proof of Proposition 2.1

To see how the above proposition can be proved, consider the non-exploratory algorithm $\mathcal{A}$ and let $\hat{S}$ be the surface for which one of the conditions of Definition 2.6 is not satisfied.

Now suppose that $\mathcal{A}$ solves the path planning problem. Then for any points $\mathbf{s}, \mathbf{t}$ with $\mathbf{s} \in \Re^3 - V_{\hat{S}}$ and $\mathbf{t} \in V_{\hat{S}}$ the path generated by $\mathcal{A}$ is of finite length. We proceed by deforming $\hat{S}$ into a new surface that contains $\mathbf{t}$ and considering the paths generated by $\mathcal{A}$ on this new surface. Intuitively, the idea of the proof is to show that any non-exploratory algorithm that plans a path from $\mathbf{s}$ to $\mathbf{t}$ must be able to "foresee" any possible deformation of the surface that would make the surface contain $\mathbf{t}$. This requires the generated paths to have unbounded length, therefore contradicting the correctness of the algorithm.

The next subsection makes formal the notion of deforming the surface in the environment. We then consider the problem of planning the path between $\mathbf{s}$ and an unreachable point $\mathbf{t}$ on the surface $\hat{S}$. Because the algorithm is deterministic, Lemma A.1 and its extensions proved in Section A.2 show that we can appropriately deform the surface without affecting the initial portion of the automaton's path, even though after this deformation process $\mathbf{t}$ becomes reachable. Lemma 3.1, whose proof we omit due to lack of space, then shows that the length of this initial portion of the path is unbounded.

In order to avoid ambiguities concerning the environment in which α operates, we write α_S for the automaton operating in the environment that contains surface S and, similarly, write p_S for its path.

A.1 Deformations of $\hat{\mathbf{S}}$

Let $\gamma : [0,1] \to \hat{S}$ be a simple closed curve on $\hat{S}$ and let $C = \gamma([0,1])$. We only consider curves C that bound an open region R_C of $\hat{S}$ homeomorphic to a disk.[4]

Definition A.1 Given any tuple $(\hat{S}, C, \mathbf{t})$ we define the deformation $\hat{S}_C^{\mathbf{t}}$ of $\hat{S}$ with respect to C and $\mathbf{t}$ as

$$\hat{S}_C^{\mathbf{t}} = \left(f_C^{\mathbf{t}} \cup i|_{\hat{S}-R_{\hat{S}}}\right)(\hat{S}) \tag{1}$$

where

i	is the identity function in $\Re^3$
$f_C^{\mathbf{t}} : \overline{R_C} \to \overline{R'_C}$	is a homeomorphism with $f\|_C = i\|_C$, $t \in f_C^{\mathbf{t}}(R_C)$, and $R'_C \subset V_{\hat{S}}$.
$\overline{R_C}, \overline{R'_C}$	are the closures of R_C and R'_C, respectively.

Since $\mathbf{t}$ is in the connected set $V_{\hat{S}}$, it follows that such a homeomorphism $f_C^{\mathbf{t}}$ exists, i.e., we can deform R_C in the above fashion without altering its genus, producing any self-intersections, or intersections with $\hat{S} - R_C$. Also, from the Glueing lemma [16] it follows that both $(f_C^{\mathbf{t}} \cup i|_{\hat{S}-R_{\hat{S}}})$ and its inverse are homeomorphisms.

For any fixed point $\mathbf{t} \in V_{\hat{S}}$, $f_C^{\mathbf{t}}$ allows us to associate a unique surface $\hat{S}_C^{\mathbf{t}}$ with each simple closed curve C bounding a disk in $\hat{S}$, such that $\hat{S}_C^{\mathbf{t}}$ contains $\mathbf{t}$. In the following we fix $\mathbf{t}$ and omit the superscript in $\hat{S}_C^{\mathbf{t}}$. Intuitively, $\hat{S}_C$ is created by producing a dent in $\hat{S}$ so that the new surface contains $\mathbf{t}$.

A.2 The Main Lemma

The main idea of the proof of Proposition 2.1 lies in the following simple observation. Since algorithm $\mathcal{A}$ is deterministic, any deformations applied to regions of $\hat{S}$ that were not sensed

[4]If $\hat{S}$ is homeomorphic to a sphere, then without loss of generality assume that as point $\gamma(t)$ moves on C by increasing t, the region R_C is to the left of $\gamma(t)$.

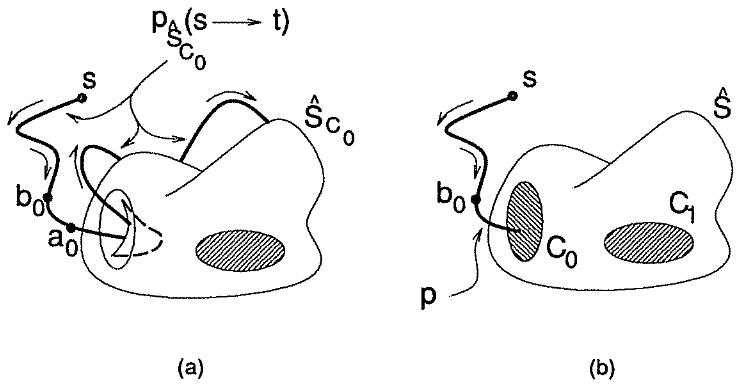

Figure 3:

along the path already traced by the automaton will not affect the execution of $\mathcal{A}$ (i.e., the path of the automaton) up to the current position of the automaton.

We make these ideas concrete as follows. Let C_0, C_1 be simple closed curves bounding disks R_{C_0}, R_{C_1} on $\hat{S}$. We consider the actions of three automata, $\alpha_{\hat{S}}, \alpha_{\hat{S}_{C_0}}$, and $\alpha_{\hat{S}_{C_1}}$. We assume that they use the same algorithm to plan their motions and the same sensing function but live in three different spaces, namely the sets $\Re^3 - V_{\hat{S}}, \Re^3 - V_{\hat{S}_{C_0}}$, and $\Re^3 - V_{\hat{S}_{C_1}}$ respectively.

Let $p_{\hat{S}_{C_0}}(\mathrm{s} \rightarrow \mathrm{t})$ be the path from s to t traced by $\alpha_{\hat{S}_{C_0}}$ in $\Re^3 - V_{\hat{S}_{C_0}}$ (Figure 3). Suppose $\mathfrak{S}(\mathrm{s}) \cap \overline{R'_{C_0}} = \emptyset$ (i.e., the automaton $\alpha_{\hat{S}_{C_0}}$ cannot sense the deformed portion of $\hat{S}$ from position s), and let a_0 be the first point on this path for which $\mathfrak{S}_{\hat{S}_{C_0}}(a_0) \cap \overline{R'_{C_0}} \neq \emptyset$. Since $s \in \Re^3 - V_{\hat{S}}$ the initial portion p of this path will be contained in $\Re^3 - V_{\hat{S}}$.

Now consider p as a path in $\Re^3 - V_{\hat{S}}$. Suppose that $\mathfrak{S}(\mathrm{s}) \cap \overline{R_{C_0}} = \emptyset$. Let b_0 be the first point of p for which $\mathfrak{S}(b_0) \cap \overline{R_{C_0}} \neq \emptyset$, and let $p_{\hat{S}_{C_0}}(\mathrm{s} \rightarrow \mathbf{b}_0)$ be the initial segment of p up to and including point b_0. We now have the following lemma:

Lemma A.1 *If $\overline{R_{C_1}} \cap \mathcal{M}(p_{\hat{S}_{C_0}}(\mathrm{s} \rightarrow \mathbf{b}_0)) = \emptyset$, the path $p_{\hat{S}_{C_1}}(\mathrm{s} \rightarrow \mathrm{t})$ traced by α_{C_1} contains $p_{\hat{S}_{C_0}}(\mathrm{s} \rightarrow \mathbf{b}_0)$.*

Proof. Consider the input available to algorithm $\mathcal{A}$ at b_0 after planning path $p_{\hat{S}_{C_0}}(\mathrm{s} \rightarrow \mathbf{b_0})$ for automaton α_{C_0}. This consists of the coordinates of $\mathbf{s}, \mathbf{t}$ and of all points in $\mathcal{M}(p_{\hat{S}_{C_0}}(\mathrm{s} \rightarrow \mathbf{b_0})$.

We now apply the following deformations on S_{C_0}:

1. Deform R'_{C_0} into R_{C_0} using $(f^{\mathbf{t}}_{C_0})^{-1}$.

2. Deform R_{C_1} into R'_{C_1} using $f^{\mathbf{t}}_{C_1}$.

First note that by definition both R_{C_0} and R'_{C_0} have no points in common with $\mathcal{M}(p_{\hat{S}_{C_0}}(\mathrm{s} \rightarrow \mathbf{b_0})$. Hence the execution of the algorithm is not affected by the first deformation step applied to $\hat{S}_{C_0}$. Second, note that by the assumptions of the lemma R_{C_1} and $\mathcal{M}(p_{\hat{S}_{C_0}}(\mathrm{s} \rightarrow \mathbf{b_0}))$ have no points in common.

The first deformation step deforms $\hat{S}_{C_0}$ into $\hat{S}$. We can therefore conclude that the paths generated by the algorithm for $\alpha_{\hat{S}_{C_0}}$ and $\alpha_{\hat{S}}$ are identical up to and including point b_0.

Now, by the definition of the deformation operation, R'_{C_1} is contained in $V_{\hat{S}}$. Since $\mathcal{M}(p_{\hat{S}}(\mathrm{s} \rightarrow \mathbf{b_0}))$ has no points in common with either $V_{\hat{S}}$ or R_{C_1}, the second deformation step cannot affect $\mathcal{A}$ at least until the automaton $\alpha_{\hat{S}_{C_1}}$ has traced $p_{\hat{S}_{C_1}}(\mathrm{s} \rightarrow \mathbf{b_0})$. $\square$

We now apply Lemma A.1 to a sequence of simple closed curves $(C)_n$ bounding a disk in $\hat{S}$ in order to show that the path generated by $\mathcal{A}$ on $\hat{S}$ will pass through enough points to make its length unbounded.

Note that for each simple closed curve C_n in the sequence, we can assign a point $b_n \in (\Re^3 - V_{\hat{S}}) \cap (\Re^3 - V_{\hat{S}_{C_n}})$ such that $p_{\hat{S}_{C_n}}(\mathrm{s} \rightarrow \mathbf{b_n}) \subset \Re^3 - V_{\hat{S}}$ and $\mathcal{M}(p_{\hat{S}_{C_n}}(\mathrm{s} \rightarrow \mathbf{b_n}))$ has no points in common with $\overline{R_{C_n}}$.

Definition A.2 (Independent sequences $(C)_n$) We call a sequence $(C)_n$ of simple closed curves bounding disks on $\hat{S}$ independent iff the following condition holds for all n:

$$\overline{R_{C_n}} \cap \left[\bigcup_{m=1}^{n-1} \mathcal{M}(p_{\hat{S}_{C_m}}(\mathrm{s} \to \mathbf{b_m})) \right] \neq \emptyset \tag{2}$$

Lemma A.2 *Let $(C)_n$ be an independent sequence of curves in $\hat{S}$. Then for any $n > 0, p_{\hat{S}_{C_n}}(\mathrm{s} \to \mathbf{t})$ contains all $p_{\hat{S}_{C_m}}(\mathrm{s} \to \mathbf{b_m}), m \leq n$.*

Proof. By induction on n, applying Lemma A.1. □

The following lemma makes explicit the connection between $p_{\hat{S}}(\mathrm{s} \to \mathbf{t})$ and $p_{\hat{S}_{C_n}}(\mathrm{s} \to \mathbf{t})$.

Corollary A.1 *Let $(C)_n$ be an independent sequence of curves. Then for any $n \geq 0, p_{\hat{S}}(\mathrm{s} \to \mathbf{t})$ contains all paths $p_{\hat{S}_{C_m}}(\mathrm{s} \to \mathbf{b_m}), m \leq n$.*

Proof. Use Lemma A.2 to show that $p_{\hat{S}_{C_n}}(\mathrm{s} \to \mathbf{t})$ contains all $p_{\hat{S}_{C_m}}(\mathrm{s} \to \mathbf{b_m}), m \leq n$.

Then apply deformation $(f^{\mathbf{t}}_{\hat{S}_{C_n}})^{-1}$ to R'_{C_k} so that $\hat{S}_{C_k}$ deforms into $\hat{S}$. Because of the independence assumption and the definition of b_n, the input to algorithm $\mathcal{A}$ upto and including point b_n will not be affected by this deformation. Therefore the path traced by the automata $\alpha_{\hat{S}}$ and $\alpha_{\hat{S}_{C_n}}$ will be identical upto and including point b_n. □

A.3 Proof of Proposition 2.1

Corollary A.1 shows that if we can construct an independent sequence of curves $(C)_n$ then $p_{\hat{S}}(\mathrm{s} \to \mathbf{t})$ contains the initial portions all paths $p_{\hat{S}_{C_m}}(\mathrm{s} \to \mathbf{b_m}), m \leq n$. Proposition 2.1 in now proved by the following lemma which shows that we can find such a sequence so that the path generated by $\mathcal{A}$ in the environment containing $\hat{S}$ is of infinite length:

Lemma A.3 *There exists an independent sequence of curves $(C)_n$ such that for any positive $M > 0$ there is an integer n_M such that*

$$\mathrm{length}(p_{\hat{S}_{C_n}}(\mathrm{s} \to \mathbf{b_n})) > M \tag{3}$$

Proof. Let C_0 be a simple closed curve bounding a disk in $\hat{S}$. To create the sequence $(C)_n$ we choose C_n to be a simple closed curve bounding a disk in $\hat{S} - \bigcup_{m=1}^{n-1}[\mathcal{M}(p_{C_m}(\mathbf{s} \to \mathbf{b_m})) \cap \hat{S}]$.

The infinite sequence $(C)_n$ is well-defined because it is always possible to select such a curve C_n. To see this, first observe that $p_{C_m}(\mathbf{s} \to \mathbf{a_m})$ contains $p_{C_{m-1}}(\mathbf{s} \to \mathbf{a_{m-1}})$ and hence we have

$$\hat{S} - \bigcup_{m=0}^{n-1} \left[\mathcal{M}(p_{C_m}(\mathbf{s} \to \mathbf{b_m})) \cap \hat{S}\right] \tag{4}$$

$$= \bigcap_{m=0}^{n-1} \left[\overline{\mathcal{M}(p_{C_m}(\mathbf{s} \to \mathbf{b_m}))} \cap \hat{S}\right] \tag{5}$$

$$= \hat{S} - \left[\mathcal{M}(p_{C_{n-1}}(\mathbf{s} \to \mathbf{a_{n-1}})) \cap \hat{S}\right] \tag{6}$$

$$= \overline{\mathcal{M}(p_{C_n}(\mathbf{s} \to \mathbf{b_n}))} \cap \hat{S} \tag{7}$$

Now note that algorithm $\mathcal{A}$ is not exploratory and $\hat{S}$ was chosen so that for any finite path p generated by $\mathcal{A}$ there exists an open set U containing no points in common with $\mathcal{M}(p)$. So we can choose C_n to be a simple closed curve bounding a disk in U.

Finally, we need to show that we can choose the curves C_n in the above manner so that the length of the path connecting points $\mathbf{s}$ and $(b)_n$ is unbounded. It suffices to show that the polygonal path connecting points $(b)_n$ is unbounded, i.e., that

$$\sum_{m=1}^{n_M} d_2(b_m, b_{m-1}) > M \tag{8}$$

where d_2 is the Euclidean distance metric in $\Re^3$.

Suppose no such sequence exists. This implies that $\sum_{n=1}^{\infty} d_2(b_n, b_{n-1})$ is bounded for all sequences $(C)_n$. Let $(C)_n$ be any such sequence. Then we have

$$\lim_{n\to\infty} \sum_{k=2}^{n} d_2(b_n, b_{n-1}) = L < \infty \tag{9}$$

since it is an increasing sequence bounded from above. Let b_∞ be the limit of the sequence $(b)_n$.

Now the curve C_n is chosen from the set $\bigcap_{m=1}^{n-1} \overline{(\mathcal{M}(p_{C_m}(\mathbf{s} \to \mathbf{t})) \cap \hat{S})}$. Curves $(C)_n$ are chosen to lie in a sequence of subset intersections

$$\bigcap_{m=0}^{1} \left[\overline{\mathcal{M}(p_{C_m}(\mathbf{s} \to \mathbf{b_m}))} \cap \hat{S}\right] \supset \cdots \supset \bigcap_{m=0}^{n} \left[\overline{\mathcal{M}(p_{C_m}(\mathbf{s} \to \mathbf{b_m}))} \cap \hat{S}\right] \supset \cdots$$

none of which is the empty set. Therefore, the limit of the sequence exists and is a set $\mathcal{M}_\infty$.

It follows that $\mathcal{M}_\infty$ does not contain any open sets. But now the path traced by the automaton up to and including the limit point b_∞ is finite and all but a degenerate set of points have been sensed by the automaton. However, this contradicts both conditions of Definition 2.6 since $\mathcal{A}$ was assumed non-exploratory. □

Proof (Proposition 2.1). Proposition 2.1 now follows directly from Lemma A.3. The second part of the proposition is proved by this lemma by setting $S_M = \hat{S}_{C_n}$ for an appropriately chosen n. The first part now directly follows, since from Corollary A.1 we have

$$p_{\hat{S}}(\mathbf{s} \to \mathbf{t}) > \sup_{n \geq 0}\{\text{length}(p_{C_n}(\mathbf{s} \to \mathbf{t}))\} = \infty \tag{10}$$

□

B Proof of Lemma 3.1

Since Steps 1-3 are each executed once during the loop of $\mathcal{A}_{3D}$, it suffices to show that Step 1 is executed a finite number of times, and that if $\mathbf{t}$ is reachable, it is reached at Step 1.

Suppose Step 1 is executed when the robot is at a position $\mathbf{s}$ from which $\mathbf{t}$ is not Π-reachable. Let y be the point of minimum distance on curve l (defined in Step 1) to $\mathbf{t}$. We first show the following lemma:

Lemma B.1 *Let L be the half-open line segment connecting y to $\mathbf{t}$ and containing $\mathbf{t}$ but not y. If L is not contained in the open volume bounded by S, the distance between y and $\mathbf{t}$ strictly decreases between successive executions of Step 1.*

Proof. Suppose that L is not contained in the open volume bounded by S, and let w be the point of intersection of L with $S \cap \Pi$ that is closest to y. We first show that this intersection is not empty.

Suppose the intersection is empty. This implies that L must lie completely outside the volume bounded by S. This, however, contradicts the definition of y since $\mathbf{t}$ would be Π-reachable in this case.

Now, since w is a point on S, it will become sensed during the exploration of S. Hence, the point x sensed at Step 2 will have a distance from $\mathbf{t}$ at most equal to the distance of w from $\mathbf{t}$. This implies that the algorithm will not terminate after performing Step 2. Furthermore, algorithm BUG1 guarantees that the point y' selected during the subsequent execution of Step 1 will have distance from $\mathbf{t}$ at most equal to the distance of x from $\mathbf{t}$. $\square$

We can now prove Lemma 3.1 by showing that Steps 1-3 are executed a finite number of times as follows:

Proof (Lemma 3.1). Consider the closed curves bounding the intersection $S \cap \Pi$. The curve l will correspond to one of these curves. Furthermore, for each curve l we can associate a distance d_l corresponding to the distance from the curve to $\mathbf{t}$. From Lemma B.1 we know that after Step 2 is executed the distance to $\mathbf{t}$ will be less than d_l, where l is the curve selected in Step 1.

To show that Step 1 is executed a finite number of times it suffices to show that only a finite number of distinct curves l can be selected, i.e., that only a finite number of curves l are contained in a disk of radius d_l, for any l.

From our assumptions about the environment it follows that each disk D of finite radius can intersect only a finite number of obstacles. Furthermore, only a finite number of closed connected regions are contained in the intersection of D with an obstacle. It follows that the total number of regions contained in $D \cap S$ is finite and, therefore, the number of curves bounding these regions is also finite.

Since Steps 1-3 are executed a finite number of times we can conclude that the algorithm terminates if $\mathbf{t}$ is not reachable. Now, suppose that $\mathbf{t}$ is reachable and that the algorithm terminates before $\mathbf{t}$ is reached. This can only occur after performing Step 2 of the algorithm. Let y be the point defined in Step 1 of the cycle of the algorithm in which the algorithm terminates. Since $\mathbf{t}$ is reachable, it follows that the half-open line segment connecting y to $\mathbf{t}$ is not contained in the volume bounded by S. This leads to a contradiction, because from Lemma B.1 the algorithm cannot not terminate at Step 2. $\square$