New Heuristic Algorithms for Efficient Hierarchical Path Planning

David Zhu and Jean-Claude Latombe

Abstract—**One of the ultimate goals of robotics research is to create autonomous robots. Progress toward this goal requires advances in many domains, including automatic motion planning. The "basic problem" in motion planning is to construct a collision-free path for a moving object among fixed obstacles. In this paper, we consider one of the most popular approaches to path planning: hierarchical approximate cell decomposition. This approach consists of constructing successive decompositions of the robot's configuration space into rectangloid cells and searching the connectivity graph built at each level of decomposition for a path. Despite its conceptual simplicity, an efficient implementation of this approach raises many delicate questions that have not yet been addressed. The major contributions of this paper are 1) a novel approach to cell decomposition based on "constraint reformulation" and 2) a new algorithm for hierarchical search with a mechanism for recording failure conditions. We have implemented these algorithms in a path planner and experimented with this planner on various examples (some of which are reported in this paper). These experiments show that our planner is significantly (approximately 10 times) faster than previous planners based on the same general approach.**

I. Introduction

ONE of the ultimate goals of robotics research is to create autonomous robots. Such robots will accept high-level descriptions of tasks and will execute them without further human intervention. The input descriptions will specify what the user wants done rather than how to do it. Progress toward this goal requires advances in many domains, including automatic reasoning, perception, and real-time control. One of the key topics in reasoning is motion planning. It is aimed at providing robots with the capability of deciding what motion commands to execute in order to achieve specified arrangements of physical objects. During the last ten years, it has emerged as a major research area [21], [28].

The "basic problem"—planning a collision-free path for a moving object among fixed obstacles—has attracted a lot of attention. Several approaches have been proposed, e.g., exact cell decomposition [15], [27], approximate cell decomposition [5], retraction [23], potential field [16], [17], [26]. Most of the existing methods, however, still lack efficiency, reliability, or both.

In this paper, we consider one of the most popular approaches to motion planning: hierarchical approximate cell decomposition. It was introduced in [5], with subsequent contributions by other authors (e.g., [10], [13], [19]). We propose a set of new algorithms for constructing more efficient path planners based on this approach. We have implemented these algorithms in a planner and experimented with them on a variety of examples. Taking into account the relative speed of the computers, our planner is significantly (approximately 10 times) faster than previous planners using the same approach. Like other path planners based on cell decomposition, our planner generates a sequence of cells, which we call a "channel." Hence, the robot does not commit itself to a single path at planning time so that it can adapt its path at execution time, for instance, to optimize dynamic behavior and to avoid unexpected obstacles [8].

The hierarchical approximate cell decomposition approach consists of decomposing the configuration space of the moving object (the robot) into rectangloid cells at successive levels of approximation. Cells are classified as EMPTY or FULL depending on whether they lie entirely outside or entirely inside the obstacles. If they are neither EMPTY nor FULL, they are labelled MIXED. At each level of approximation, the planner searches the graph of the adjacency relation among the cells for a sequence of adjacent EMPTY cells connecting the initial configuration of the robot to the goal configuration. If no such sequence is found, it decomposes some MIXED cells into smaller cells, labels them appropriately, and searches again for a sequence of EMPTY cells. The process ends when a solution has been found, or it is guaranteed that no solution can be found, or MIXED cells are smaller than some prespecified size.

Most motion planning approaches have their own advantages and drawbacks, which have to be weighed in the function of the context in which they are considered. Nevertheless, the hierarchical cell decomposition approach presents a rather unique combination of attractive features. It is relatively easy to implement even when the moving object can both translate and rotate. It is reasonably efficient when the number of degrees of freedom of the moving object is small, and parallelization is possible for achieving better performances. It is complete under reasonable assumptions and "resolution-complete" otherwise. Finally, as mentioned above, it produces a channel rather than a single path.

However, despite the conceptual simplicity of the main two steps of the approach, i.e., cell decomposition and graph searching, their efficient implementation raises delicate questions not thoroughly addressed in previous publications. Although we were implementing and experimenting with a planner for a mobile robot, we found out that efficiency can be sharply increased by shifting from naive answers to these questions to more sophisticated ones. In this paper, we present in detail a novel approach to cell decomposition based on "constraint reformulation" and a new algorithm for hierarchical search with a mechanism for recording failure conditions. We also report on our experimental results with the implemented planner.

The problem of decomposing a MIXED cell is to maximize the volume of the EMPTY and FULL cells resulting from the decomposition in order to make it possible to find a path (or to discover the absence of a path) as quickly as possible. In

Manuscript received August 1, 1989; revised June 21, 1990. This work was supported by DARPA under Contract DAAA21-89-C0002 (Army), by the Center of Integrated Systems (CIS) and by the Center for Integrated Facility Engineering (CIFE).

The authors are with the Robotics Laboratory, Computer Science Department, Stanford University, Stanford, CA 94305.

IEEE Log Number 9040140.

1042-296X/91/0200-0009$01.00

particular, the blind 2^n-tree (e.g., quadtree, octree) decomposition technique has the drawback of decomposing many MIXED cells into smaller MIXED cells. We propose a new "constraint reformulation" technique, which provides better results than earlier decomposition techniques. It consists of approximating the obstacles intersecting with the MIXED cell to be decomposed by a collection of rectangloids and computing the complement of these rectangloids in the cell. Two types of approximation are used: "bounding" and "bounded" approximations. The first is used to produce EMPTY cells, the second to produce FULL cells.

The problem of graph searching is to take advantage of unsuccessful search work done at prior levels of approximation since most of the search graph remains the same from one level to the next (only the portions of the graph that correspond to decomposed MIXED cells are modified). We propose new search techniques based on both an appropriate representation of the search graph and the recording of failure conditions. These techniques, which are inspired from those implemented in dependency directed backtracking systems [18], [30] avoid the path planner to run into the same mistakes several times.

II. Background and Overview

A. Configuration Space

Let us consider a rigid object $\mathcal{A}$ moving in an Euclidean workspace $\mathcal{W} = \boldsymbol{R}^N$ among fixed obstacles $\mathcal{B}_i$, $i = 1, \cdots, q$. Both $\mathcal{A}$ and the $\mathcal{B}_i$'s are closed regions in $\boldsymbol{R}^N$. A Cartesian coordinate frame $\mathcal{F}_{\mathcal{W}}$ is embedded in $\mathcal{W}$. Another frame $\mathcal{F}_{\mathcal{A}}$ is embedded in $\mathcal{A}$. The origin $\mathcal{O}_{\mathcal{A}}$ of $\mathcal{F}_{\mathcal{A}}$ is called the reference point of $\mathcal{A}$.

A configuration of $\mathcal{A}$ is a specification of the position and orientation of $\mathcal{F}_{\mathcal{A}}$ with respect to $\mathcal{F}_{\mathcal{W}}$. The configuration space of $\mathcal{A}$ is the space $\mathcal{C}$ of all the possible configurations of $\mathcal{A}$. The unique configuration where $\mathcal{F}_{\mathcal{A}}$ and $\mathcal{F}_{\mathcal{W}}$ coincide is called the reference configuration of $\mathcal{A}$. The subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at configuration $\boldsymbol{q}$ is denoted by $\mathcal{A}(\boldsymbol{q})$.

The obstacles $\mathcal{B}_i$ map in $\mathcal{C}$ to closed regions $\mathcal{CB}_i$ (called C obstacles) defined by

$$\mathcal{CB}_i = \{\boldsymbol{q} \in \mathcal{C} / \mathcal{A}(\boldsymbol{q}) \cap \mathcal{B}_i \neq \emptyset\}.$$

The region

$$\mathcal{C}_{\text{free}} = \mathcal{C} - \bigcup_{i \in [1, q]} \mathcal{CB}_i$$

is called the free space. A collision-free path (more simply, a free path) is any continuous map $\tau: [0, 1] \rightarrow \mathcal{C}_{\text{free}}$.

In this paper, we consider the case where $\mathcal{A}$ is a two-dimensional object that translates and rotates in $\mathcal{W} = \boldsymbol{R}^2$. In this case, $\mathcal{C}$ is a manifold diffeomorphic to $\boldsymbol{R}^2 \times S^1$, where S^1 denotes the unit circle [29]. We parameterize a configuration in this manifold by a triplet (x, y, θ), where $x, y \in \boldsymbol{R}^2$ are the coordinates of $\mathcal{O}_{\mathcal{A}}$ in $\mathcal{F}_{\mathcal{W}}$, and $\theta \in [0, 2\pi)$ is the angle (modulo 2π) between the x axes of $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_{\mathcal{A}}$.

B. C Obstacles in Polygonal Case

Throughout the paper, both $\mathcal{A}$ and the $\mathcal{B}_i$'s are polygons. Under this condition, the cross section of each $\mathcal{CB}_i$ at any orientation θ is also a polygon defined by [5]

$$\mathcal{CB}_i = \mathcal{B}_i \ominus \mathcal{A}(0, 0, \theta)$$

where $\ominus$ is the symbol for the Minkowski difference. Each edge of this polygon is the locus of $\mathcal{O}_{\mathcal{A}}$ when $\mathcal{A}$ translates at fixed orientation θ in such a way that an edge (resp. a vertex) of $\mathcal{A}$ stays in contact with a vertex (resp. an edge) of $\mathcal{B}_i$ (see Fig. 1). A contact between an edge of $\mathcal{A}$ and a vertex of $\mathcal{B}_i$ is called a type-A contact. When $\mathcal{A}$ rotates slightly, the corresponding edge of $\mathcal{CB}_i$ rotates by the same angle. A contact between a vertex of $\mathcal{A}$ and an edge of $\mathcal{B}_i$ is called a type-B contact. When $\mathcal{A}$ rotates slightly, the corresponding edge of $\mathcal{CB}_i$ translates accordingly. Therefore, the C obstacle $\mathcal{CB}_i$ corresponding to an obstacle $\mathcal{B}_i$ is a three-dimensional volume bounded by patches of ruled surfaces, which we call C facets. Each C facet is generated by a straight line segment of variable length, which remains parallel to the x–y plane and either translates or rotates. A C facet created by a contact of type A (resp., type B) is called a type-A (resp., type-B) C facet. Each C facet is comprised between two limit orientations beyond which the contact that creates the C facet is no longer feasible. The geometry of $\mathcal{CB}_i$ when $\mathcal{A}$ and $\mathcal{B}_i$ are polygons is studied in depth in various publications, e.g., [2], [6], [9], [21]. If $\mathcal{A}$ and $\mathcal{B}_i$ are both convex polygons, $\mathcal{CB}_i$ is bounded by $O(n_{\mathcal{A}} n_{\mathcal{B}_i})$ C facets, where $n_{\mathcal{A}}$ and $n_{\mathcal{B}_i}$ are the number of edges in $\mathcal{A}$ and $\mathcal{B}_i$, respectively. If $\mathcal{A}$ and $\mathcal{B}_i$ are nonconvex, $\mathcal{CB}_i$ has $\Omega(n_{\mathcal{A}}^3 n_{\mathcal{B}_i}^3)$ C facets [2].

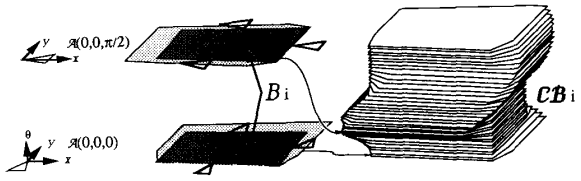


Fig. 1. C obstacle.

C. Rectangloid Decomposition

In the following, we assume, without practical loss of generality, that the range of possible values for x and y are closed intervals $[x_{\min}, x_{\max}]$ and $[y_{\min}, y_{\max}]$. We represent $\mathcal{C}$ as a closed rectangloid

$$\mathcal{K} = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [0, 2\pi] \subset \boldsymbol{R}^3$$

with the two cross sections at $\theta = 0$ and $\theta = 2\pi$ procedurally identified.

Let κ be a rectangloid, i.e., a region of the form:

$$[x_1, x_2] \times [y_1, y_2] \times [\theta_1, \theta_2] \subseteq \mathcal{K}.$$

A rectangloid decomposition of κ is a collection of rectangloids, $\{\kappa_j\}_{j=1,\cdots,n}$, such that 1) κ is equal to the union of the κ_j, i.e., $\kappa = \bigcup_{j=1}^{n} \kappa_j$, and 2) the κ_j are nonoverlapping,[1] i.e., $\forall j_1, j_2 \in [1, n]$, $j_1 \neq j_2$: $\text{int}(\kappa_{j_1}) \cap \text{int}(\kappa_{j_2}) = \emptyset$.

Each rectangloid κ_j is called a cell of the decomposition. Two cells κ_{j_1} and κ_{j_2} are adjacent iff their intersection is a set of nonzero measure in $\boldsymbol{R}^2$, i.e., a surface of nonzero area. (The intersection is computed by taking into account that the cross sections at 0 and 2π are identified.)

A cell κ_j is classified in the following ways:

- EMPTY, iff its interior $\text{int}(\kappa_j)$ intersects no C obstacle, i.e., $\text{int}(\kappa_j) \cap \bigcup_i \mathcal{CB}_i = \emptyset$
- FULL, iff κ_j is entirely contained in the union of the C obstacles, i.e., $\kappa_j \subset \bigcup_i \mathcal{CB}_i$
- MIXED, otherwise.

[1] Throughout this paper, we say that two closed sets are nonoverlapping iff their interiors do not intersect.

The connectivity graph associated with a decomposition of κ is the nondirected graph G defined as follows:

- The nodes of G are the EMPTY and the MIXED cells of the decomposition.
- Two nodes of G are connected by a link iff the corresponding cells are adjacent.

Given a rectangloid decomposition $\{\kappa_j\}_{j=1,\cdots,n}$ of κ, a channel is defined as a sequence $(\kappa_{\alpha_l})_{l=1,\cdots,p}$ of EMPTY and/or MIXED cells such that any two consecutive cells κ_{α_l} and $\kappa_{\alpha_{l+1}}$, $l \in [1, p-1]$ are adjacent. A channel that only contains EMPTY cell is called an E channel. A channel that contains at least one MIXED cells is called an M channel. If $(\kappa_{\alpha_l})_{l=1,\cdots,p}$ is an E channel, then any path connecting any configuration in κ_{α_1} to any configuration in κ_{α_p} and lying in $\text{int}(\bigcup_{l=1}^{p}\kappa_{\alpha_l})$ is a free path. If $(\kappa_{\alpha_l})_{l=1,\cdots,p}$ is an M channel, there may exist free paths connecting two configurations in κ_{α_1} and κ_{α_p} and lying in $\text{int}(\bigcup_{l=1}^{p}\kappa_{\alpha_l})$, but there is no guarantee that this is the case.

Given an initial configuration $q_{\text{init}} = (x_{\text{init}}, y_{\text{init}}, \theta_{\text{init}}) \in \mathscr{C}_{\text{free}}$ and a goal configuration $q_{\text{goal}} = (x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}}) \in \mathscr{C}_{\text{free}}$, the problem is to generate an E channel $(\kappa_{\alpha_l})_{l=1,\cdots,p}$ such that $q_{\text{init}} \in \kappa_{\alpha_1}$ and $q_{\text{goal}} \in \kappa_{\alpha_p}$. If such a channel is generated, a path can easily be constructed as an open polygonal line. If necessary, it can be smoothed by fitting splines curves [14].

D. Hierarchical Planning

Hierarchical path planning consists of generating an E channel by constructing successive rectangloid decompositions of $\mathscr{K}$ and searching the associated connectivity graph. Let $\mathscr{P}_i$, $i = 0, 1, \cdots$, denote the successive decompositions of $\mathscr{K}$, with $\mathscr{P}_0 = \{\mathscr{K}\}$. Each decomposition $\mathscr{P}_i$, $i > 0$ is obtained from the previous one $\mathscr{P}_{i-1}$ by decomposing one or several MIXED cells, the other cells being unchanged. Whenever a decomposition $\mathscr{P}_i$ is constructed, the corresponding cell connectivity graph G_i is searched for a channel connecting q_{init} and q_{goal}. A simple hierarchical search algorithm is as follows:

1) Generate a first decomposition $\mathscr{P}_1$ of $\mathscr{K}$. Construct the graph G_1 corresponding to this decomposition. Set i to 1.
2) Search G_i for a channel connecting q_{init} and q_{goal}. If an E channel is found, return success. If an M channel is found, go to next step; otherwise, return failure.
3) Let Π be the M channel found at Step 2. Set $\mathscr{P}_{i+1}$ to $\mathscr{P}_i$ and i to $i+1$. For every MIXED cell κ in Π, decompose κ into a set $\mathscr{P}^{\kappa}$ of smaller cells, and set $\mathscr{P}_i$ to $[\mathscr{P}_i \setminus \kappa] \cup \mathscr{P}^{\kappa}$. Go to Step 2.

This algorithm searches successive graphs until an E channel is found. Each graph G_i, $i > 1$ is obtained from G_{i-1} by expanding the MIXED cells, which belong to the M channel generated in G_{i-1}.

The search for a channel in a graph may be guided by various heuristics. If we want to generate an E channel faster, we put a large penalty on MIXED cells. If we prefer shorter channels instead, we put a larger penalty on the channel length (using some metrics). Then, although an E channel may exist in a graph G_i, the algorithm may generate a significantly shorter M channel and refine G_i accordingly. Notice that any E channel existing in G_i will continue to exist in G_j, $\forall j > i$.

Let us assume that the region occupied by the C obstacles can be expressed as the union of finitely many disjoint subregions, where each is equal to the closure of its interior. Then, the planning process outlined above can be made complete, i.e., guaranteed to terminate successfully whenever a free path exists, and to return failure otherwise, by working out some details of the algorithms appropriately. However, for an unknown problem, there is no time bound on the process since there is no lower bound on the size of the cells that may have to be generated.

The worst-case time complexity can be bounded at the expense of the completeness by imposing constraints on the decomposition of any MIXED cell κ. One possible constraint is that the total volume of the EMPTY and FULL cells extracted from κ be greater than a predefined ratio of the volume of κ; in addition, every MIXED cell resulting from the decomposition of κ that has any of its dimensions (side length) smaller than a predefined value is relabeled FULL. Another possible constraint is that every generated cell should have dimensions greater than prespecified values. The choice of the constraints on the decomposition of a MIXED cell depends on the decomposition technique that is used. If the decomposition algorithm fails to find a decomposition of κ satisfying the constraint, κ is said to be nondecomposable and is relabeled FULL.

If constraints are imposed on MIXED cell decomposition, the algorithm is no longer guaranteed to find a free path whenever one exists. However, it is "resolution-complete," i.e., if an E channel exists at the resolution determined by the constraints, it will be found.

III. Cell Decomposition

A. Issue

The subproblem considered in this section is that of generating a rectangloid decomposition $\{\kappa_j\}_{j=1,\cdots,n}$ of a given MIXED cell κ. For the efficiency of the planning process, the generated decomposition should simultaneously satisfy the following two goals:

1) The number of cells in the decomposition should be reasonably small in order to keep the size of the search graph as tractable as possible. This goal is directly related to the motivation for the overall hierarchical strategy: We want the planner to consider "details" only if it is necessary.
2) The volume of the EMPTY and the FULL cells should be large relative to the total volume of κ. This goal is aimed at reducing as quickly as possible the "uncertain area," i.e., the MIXED cells. Clearly, decomposing a MIXED cell into smaller cells would not be useful if all of these cells were MIXED.

These two goals may be conflicting since one obvious way to achieve the second goal is to produce many small cells, which is in contradiction with the first one. The "constraint reformulation" algorithm described in this section is an attempt to achieve both these goals simultaneously.

B. Previous Approaches

One simple method of decomposing a MIXED cell is to partition it into 2^m cells of equal size, where m is the dimension of configuration space (three in our case). The overall decomposition of configuration space obtained with this method is called "quadtree" when $m = 2$ and an "octree" when $m = 3$ [3]. The application of this method is reported in [13] (quadtree) and [10] (octree). The advantage of this method is that it yields a decomposition that is easily representable as a tree of degree 2^m. Its drawback is that most of the time, none or a few of the newly

generated cells are EMPTY or FULL. The method tends to produce a huge number of cells.

Another method is described in [5]. The basic idea is to consider potential cuts of the cell, score them, and choose the best. The potential cuts are chosen wherever a C surface[2] will go through a vertex of one of the new cells generated by the decomposition. The scoring function favors cuts that do not generate small cells, i.e., the cuts closer to the midpoints of each edge are preferred. The scoring function also attempts to minimize the number of C surfaces intersected by each new cell in order to reduce future computations. This method has the drawback of treating each C surface (not C facets) individually and to combine the effect of the various C surfaces in a global scoring function. Although certainly better than a 2^m tree, the resulting decomposition may still be far from optimal. It may also incorrectly[3] label a cell that intersects no C obstacle as MIXED.

The drawbacks of these previous approaches are caused by the separation of the decomposition process from the labeling process. They decompose the space more or less blindly and then label the cells by comparing them to the C obstacles. Nonoptimal decomposition yields a larger number of cells, which in turn increases both the cost of labeling the new cells and that of searching the connectivity graph.

C. Constraint Reformulation Approach

We propose a new approach for decomposing MIXED cells; this approach consists of first approximating each C obstacle intersecting the cell κ to be decomposed as a collection of nonoverlapping rectangloids. The complement of a union of rectangloids within a rectangloid region is also a union of rectangloids that can easily be computed. This yields a rectangloid decomposition of κ. We call this approach constraint reformulation because it basically consists of reformulating the constraints imposed by the C obstacles into a form directly compatible with the format of the decomposition of $\mathcal{A}$ into rectangloid cells. Even though it may cost slightly more time to decide on the decomposition than with a blind decomposition technique, there is no additional cost of labeling the cells.

Let $\kappa = [x_1, x_2] \times [y_1, y_2] \times [\theta_1, \theta_2]$ be the MIXED cell to be decomposed. For every $i \in [1, q]$, let $\mathcal{CB}_i[\kappa] = \mathcal{CB}_i \cap \kappa$.

Our planar generates and uses two types of approximation of C obstacles: bounding and bounded approximations:

- A **bounding approximation** of $\mathcal{CB}_i[\kappa]$ is a collection of nonoverlapping rectangloids $\mathcal{R}_{ik}$'s, $k = 1, \cdots, p$, with $\forall k \in [1, p]$, $\mathcal{R}_{ik} \subset \kappa$ and $\mathcal{CB}_i[\kappa] \subseteq \bigcup_{k=1,\cdots,p} \mathcal{R}_{ik}$.
- A **bounded approximation** of $\mathcal{CB}_i[\kappa]$ is a collection of nonoverlapping rectangloids $\mathcal{R}'_{ik}$'s, $k = 1, \cdots, p'$, with $\bigcup_{k=1,\cdots,p'} \mathcal{R}'_{ik} \subseteq \mathcal{CB}_i[\kappa]$.

The EMPTY cells of the decomposition of κ are obtained by computing the complement of $\bigcup_i \bigcup_k \mathcal{R}_{ik}$ in κ. The FULL cells are the $\mathcal{R}'_{ik}$'s. The MIXED cells are obtained by computing the complement of $\bigcup_i \bigcup_k \mathcal{R}'_{ik}$ in every $\mathcal{R}_{ik}$.

Fig. 2 illustrates the notion of bounding and bounded approximations in a two-dimensional space. Fig. 3(a) illustrates the rectangloid decomposition of a cell κ into EMPTY, FULL, and MIXED cells, using these two approximations. The quadtree decomposition at a similar resolution is shown in Fig. 3(b). We see that the decomposition using bounding and bounded approximations generates a much smaller MIXED area and a larger EMPTY/FULL area.

[2] A C surface is the infinite ruled surface that supports a C facet.

[3] This conservative "error" gets corrected when the decomposition proceeds deeper.

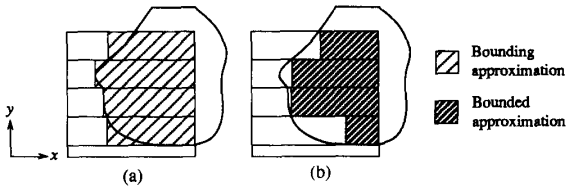


Fig. 2. Bounding and bounded approximations.

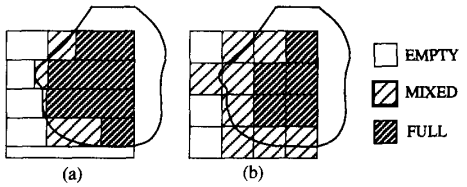


Fig. 3. Decomposition of a two-dimensional cell.

Bounding and bounded approximations have been used previously in computer graphics to simplify clipping and ray tracing [7] and to speed up the transformation and display of objects [11]. It is also used in computer vision to facilitate the computation of surface intersection [24].

D. Outline of the Algorithm

There are many ways to compute bounding and bounded approximations of a C obstacle $\mathcal{CB}$ in a cell κ. We use a "project-and-lift" method, which consists of projecting $\mathcal{CB}$ on the xy plane, next on the x or y axis, and then lifting back the projection into rectangloid regions.

1) Projection on the x-y Plane: The $[\theta_1, \theta_2]$ interval is cut into nonoverlapping subintervals $[\gamma_u, \gamma_{u+1}]$, $u = 1, \cdots, r$, $r \geq 1$, with $\gamma_1 = \theta_1$ and $\gamma_{r+1} = \theta_2$. We denote by κ^u the rectangloid $[x_1, x_2] \times [y_1, y_2] \times [\gamma_u, \gamma_{u+1}]$.

For every $u \in [1, r]$, we compute the outer projection and the inner projection of $\mathcal{CB}[\kappa^u]$ on the x-y plane. These two projections, which we respectively denote by $\mathcal{OCB}_{xy}[\kappa^u]$ and $\mathcal{ICB}_{xy}[\kappa^u]$, are defined by

$$\mathcal{OCB}_{xy}[\kappa^u] = \{(x, y) / \exists \theta \in [\gamma_u, \gamma_{u+1}] : (x, y, \theta) \in \mathcal{CB}[\kappa^u]\}$$

$$\mathcal{ICB}_{xy}[\kappa^u] = \{(x, y) / \forall \theta \in [\gamma_u, \gamma_{u+1}] : (x, y, \theta) \in \mathcal{CB}[\kappa^u]\}.$$

Clearly, we have: $\mathcal{ICB}_{xy}[\kappa^u] \subseteq \mathcal{OCB}_{xy}[\kappa^u]$.

2) Projection on the y (or x) Axis: For every $u \in [1, r]$, the interval $[x_1, x_2]$ or $[y_1, y_2]$, whichever is longer, is cut into nonoverlapping subintervals. Without loss of generality, let us assume that $[x_1, x_2]$ is subdivided. The generated subintervals are $[a_v, a_{v+1}]$, $v = 1, \cdots, s$, $s \geq 1$ with $a_1 = x_1$ and $a_{s+1} = x_2$. We denote by κ^{uv} the rectangloid $[a_v, a_{v+1}] \times [y_1, y_2] \times [\gamma_u, \gamma_{u+1}]$.

For every $v \in [1, s]$, we compute the outer projection of $\mathcal{OCB}_{xy}[\kappa^u]$ and the inner projection of $\mathcal{ICB}_{xy}[\kappa^u]$ on the y axis. They are denoted by $\mathcal{OCB}_y[\kappa^{uv}]$ and $\mathcal{ICB}_y[\kappa^{uv}]$, respectively, and are defined as follows:

$$\mathcal{OCB}_y[\kappa^{uv}] = \{y / \exists x \in [a_v, a_{v+1}] : (x, y) \in \mathcal{OCB}_{xy}[\kappa^u]\}$$

$$\mathcal{ICB}_y[\kappa^{uv}] = \{y / \forall x \in [a_v, a_{v+1}] : (x, y) \in \mathcal{ICB}_{xy}[\kappa^u]\}.$$

Both $\mathcal{OCB}_y[\kappa^{uv}]$ and $\mathcal{ICB}_y[\kappa^{uv}]$ are sets of intervals. $\mathcal{ICB}_y[\kappa^{uv}] \subseteq \mathcal{OCB}_y[\kappa^{uv}]$.

Each rectangloid $[a_v, a_{v+1}] \times [b, b'] \times [\gamma_u, \gamma_{u+1}]$, where $[b, b'] \in \mathcal{OCB}_y[\kappa^{uv}]$ (resp. $\mathcal{ICB}_y[\kappa^{uv}]$), is a rectangloid $\mathcal{R}_{ik}$ (resp. $\mathcal{R}'_{ik}$) in the bounding (resp. bounded) approximation of $\mathcal{CB}_i[\kappa]$. The choice of the γ_u's and the a_v's is empirical. Various heuristics can be used, but most of them seem to have limited effect on the average efficiency of the method. The only useful guideline is to keep the three dimensions of every MIXED cell approximately "homogeneous." Let δx, δy, and $\delta\theta$ be these dimensions. We say they are "homogeneous" iff $\delta x \approx \delta y \approx \rho\delta\theta$, where ρ is the maximal distance between $\mathcal{O}_\mathcal{A}$ and the points in the boundary of $\mathcal{A}$.

In the next two subsections, we describe the computation of the outer and inner projection of $\mathcal{CB}[\kappa^u]$ onto the x–y plane. We propose two different methods. The first, called the "projection" method, consists of computing the projection of the surface of $\mathcal{CB}$ comprised between γ_u and γ_{u+1} and clipping the subset of the projections contained in $[x_1, x_2] \times [y_1, y_2]$. The second method, which is called the "swept-area" method, consists of computing the "outer" and the "inner" swept areas of $\mathcal{A}$ when it rotates around the reference point from orientation γ_u to orientation γ_{u+1} and growing the obstacle $\mathcal{B}$ by these two areas. The projection method is preferred when the interval $[\gamma_u, \gamma_{u+1}]$ is small. The swept-area method runs significantly faster when this interval is large; however, it does not exactly compute $\mathcal{ICB}_{xy}[\kappa^u]$ but a conservative subset of it.

In the following, we assume that the reference point $O_\mathcal{A}$ is chosen in the interior of $\mathcal{A}$. In fact, the choice of $O_\mathcal{A}$ has some impact on the efficiency of the decomposition. The "best" location of $O_\mathcal{A}$ should minimize the area of the outer projection of $\mathcal{CB}[\kappa^u]$ on the x–y plane while maximizing the area of the inner projection so that less MIXED cells are ultimately generated. The center of the "smallest enclosing circle" of $\mathcal{A}$ is probably close to minimizing the area of the outer projection, whereas the center of the "largest enclosed circle" of $\mathcal{A}$ may be close to maximizing the area of the inner projection. A compromise between these two locations is in general necessary since they do not coincide.

E. Projection Method

1) Principle: Let us first assume that both $\mathcal{A}$ and $\mathcal{B}$ are convex polygons. Consider the point (x_0, y_0) in the x–y plane and the segment $\{(x_0, y_0, \theta)/\theta \in [\gamma_u, \gamma_{u+1}]\}$ above this point. If the segment pierces a C facet of $\mathcal{CB}$, then (x_0, y_0) is in $\mathcal{OCB}_{xy}[\kappa^u]$ but not in $\mathcal{ICB}_{xy}[\kappa^u]$. If it pierces no C facet, then either (x_0, y_0) is not in $\mathcal{OCB}_{xy}[\kappa^u]$, or it is in $\mathcal{ICB}_{xy}[\kappa^u]$. The segment $\{(x_0, y_0, \theta)/\theta \in [\gamma_u, \gamma_{u+1}]\}$ pierces a C facet e iff (x_0, y_0) lies in the projection of e in the x–y plane.

We show below that each C facet comprised between γ_u and γ_{u+1} projects on the x–y plane according to a generalized polygon.[4] The projection of the boundary of $\mathcal{CB}$ that is comprised between γ_u and γ_{u+1} is the union of generalized polygons, each being the projection of a C facet. This union is a "donut" shaped region (see Fig. 4) with an outer boundary Γ_1 and an inner boundary Γ_2. The region bounded by Γ_1 is the outer projection $\mathcal{OCB}_{xy}[\kappa^u]$. The region bounded by Γ_2 is the inner projection $\mathcal{ICB}_{xy}[\kappa^u]$. Therefore, the projection method consists of 1) projecting all the C facets (to the extent they are contained in the interval $[\gamma_u, \gamma_{u+1}]$) on the x–y plane, 2) clipping the union of the projections by the rectangle $[x_1, x_2] \times [y_1, y_2]$ and, within this rectangle, tracking Γ_1 and Γ_2. The projection method may be made faster by associating with each interval $[\gamma_u, \gamma_{u+1}]$ the list of all the C obstacles having a nonempty intersection with the interval and for each of these C obstacles the list of the C facets that intersect the interval. If the interval gets decomposed further, only these C facets have to be considered. The projection method is efficient when the number of C facets that have to be projected is small, that is, when the interval $[\gamma_u, \gamma_{u+1}]$ is small.

[4] A generalized polygon is a two-dimensional region homeomorphic to the closed disc whose boundary consists of straight and circular edges.

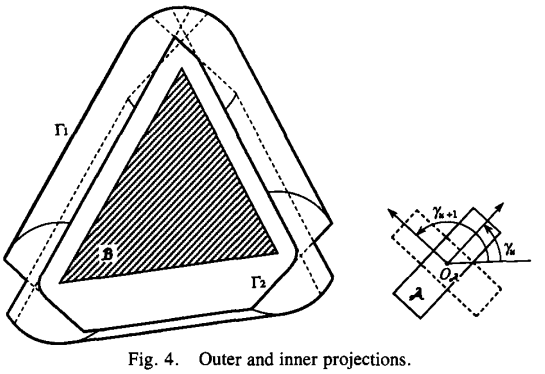


Fig. 4. Outer and inner projections.

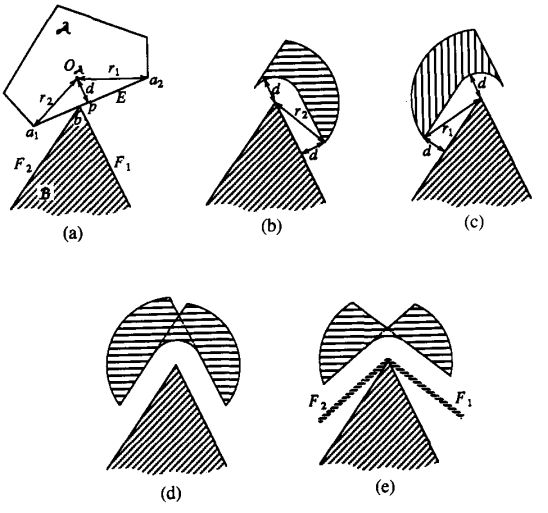


Fig. 5. Projection of a type-A C facet.

2) Projection of Type-A C Facet: Let e_A be a C facet of Type A comprised between the limit orientations ϕ_1 and ϕ_2. The contact that generates e_A is illustrated in Fig. 5(a). It occurs between an edge E of $\mathcal{A}$ and a vertex b of $\mathcal{B}$. E connects two vertices of $\mathcal{A}$: a_1 and a_2. b is the extremity of two edges of $\mathcal{B}$: F_1 and F_2. $O_\mathcal{A}$ projects on the supporting line of E at the point p. We assume below that p is located between a_1 and a_2. (The case where p is outside the segment $\overline{a_1 a_2}$ is treated in a very similar fashion.) The orientation ϕ_1 (resp. ϕ_2) is achieved when the edge E is aligned with the edge F_1 (resp. F_2). Assuming that $[\phi_1, \phi_2] \subseteq [\gamma_u, \gamma_{u+1}]$, the projection of e_A on the x–y plane is shown in Fig. 5(d). It is the union of two regions shown in Figs. 5(b) and (c).

The region in Fig. 5(b) is the locus of $O_\mathcal{A}$, when $\mathcal{A}$ translates and rotates while the edge segment $\overline{a_1 p}$ stays in contact with the vertex b. This region is bounded by two circular arcs and two

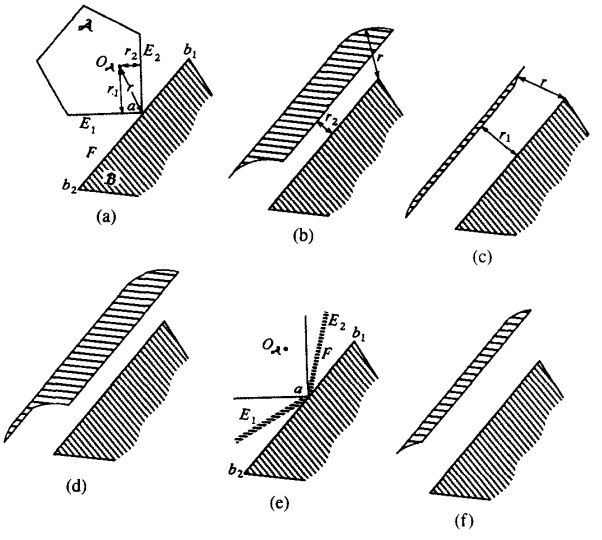


Fig. 6. Projection of a type-B C facet.

straight segments. The two arcs are centered at b. The smaller one is the locus of $O_{\mathcal{A}}$ when p coincides with b and $\mathcal{A}$ rotates from ϕ_1 to ϕ_2. The larger arc is the locus of $O_{\mathcal{A}}$ when a_1 coincides with b. The straight segments are the loci of $O_{\mathcal{A}}$ when $\mathcal{A}$ translates at fixed orientations ϕ_1 and ϕ_2 from the position where p and b coincide with the position where a_1 and b coincide. The region in Fig. 5(c) is the locus of $O_{\mathcal{A}}$, when $\mathcal{A}$ translates and rotates while the edge segment $\overline{pa_2}$ stays in contact with the vertex b.

The case where $[\gamma_u, \gamma_{u+1}] \subset [\phi_1, \phi_2]$ is treated in the same way by drawing fictitious edges F_1 and F_2 from vertex b (see Fig. 5(e)) so that $\mathcal{A}$'s orientation is γ_u (resp. γ_{u+1}) when E is aligned with F_1 (resp. F_2).

3) Projection of a Type-B C Facet: Let e_B be a C facet of Type B comprised between the limit orientations ϕ_1 and ϕ_2. The contact that generates e_B is illustrated in Fig. 6(a). It occurs between a vertex a of $\mathcal{A}$ and an edge F of $\mathcal{B}$. a is the extremity of two edges of $\mathcal{A}$: E_1 and E_2. F connects two vertices of $\mathcal{B}$: b_1 and b_2. Since we assumed $\mathcal{A}$ to be convex, $O_{\mathcal{A}}$ lies within the convex angular sector bounded by the two half lines supporting E_1 and E_2. The orientation ϕ_1 (resp. ϕ_2) is achieved when the edge E_2 (resp. E_1) is aligned with the edge F. Assuming that $[\phi_1, \phi_2] \subseteq [\gamma_u, \gamma_{u+1}]$, the projection of e_B on the x–y plane is shown in Fig. 6(d). It is the union of two regions shown in Figs. 6(b) and (c).

Let ψ be the orientation of $\mathcal{A}$ when a lies in F and the segment $\overline{aO_{\mathcal{A}}}$ is perpendicular to F. The region in Fig. 6(b) is the locus of $O_{\mathcal{A}}$ when $\mathcal{A}$ translates and rotates with a staying in F and the orientation θ ranging over $[\phi_1, \psi]$. (If $\psi < \phi_1$, the region is empty.) This region is bounded by two circular arcs and two straight segments. The two arcs are centered at b_1 and b_2, respectively, and have the same radius equal to the distance between a and $O_{\mathcal{A}}$. The region in Fig. 6(c) is the locus of $O_{\mathcal{A}}$ when $\mathcal{A}$ translates and rotates with a staying in F and the orientation θ ranging over $[\psi, \phi_2]$.

The case where $[\gamma_u, \gamma_{u+1}] \subset [\phi_1, \phi_2]$ is treated in the same way by drawing fictitious edges E_1 and E_2 from vertex a (see Figs. 6(e) and (f)) so that $\mathcal{A}$'s orientation is γ_u (resp. γ_{u+1}) when E_1 (resp E_2) is aligned with F.

4) Computation of $\mathcal{OCB}_{xy}[\kappa^u]$ and $\mathcal{ICB}_{xy}[\kappa^u]$: The projection of every C facet is a generalized polygon with a small number of edges—two to four straight segments and two or three circular arcs. The union of all the generalized polygons forms a "donut" shaped region (see Fig. 4). The contours Γ_1 and Γ_2 can be extracted by a sweep-line technique [25] and clipped by the rectangle $[x_1, x_2] \times [y_1, y_2]$. The projections $\mathcal{OCB}_y[\kappa^{uv}]$ and $\mathcal{ICB}_y[\kappa^{uv}]$ are easily computed by determining the points of Γ_1 and Γ_2 at abscissae a_v, $v = 1, \cdots, s+1$ and the extremal points of Γ_1 and Γ_2 within each interval $[a_v, a_{v+1}]$, $v = 1, \cdots, s$. This computation can be done during line sweeping. If the interval $[x_1, x_2]$ is decomposed into subintervals, the sweep line has to be parallel to the x axis; otherwise, it should be parallel to the y axis.

The overall sweep-line process takes time $O((n+c)\log n)$, where n is the number of C facets that intersect with the interval $[\gamma_u, \gamma_{u+1}]$, and c is the number of intersections of the generalized polygons. We know that $n \leq 2n_{\mathcal{A}}n_{\mathcal{B}}$. On the other hand, $c \in O(n^2)$, but it is usually much smaller.

5) Generalization: If $\mathcal{A}$ is a nonconvex polygon that can be decomposed into convex components such that the interiors of all these components have a nonempty intersection, the reference point can be selected within this intersection, and the above method directly applies to each component taken separately.

If $\mathcal{A}$ is nonconvex and cannot be represented as the union of overlapping convex components (for instance, it is a $\sqcup$-shaped object), the above method can still be applied but with some changes. Since the reference point will be outside some of the convex components, the shape of the projection of the corresponding C facets will be different but not difficult to establish.

Avnaim and Boissonnat [2] described an algorithm of time complexity $O(n_{\mathcal{A}}^3 n_{\mathcal{B}}^3 \log n_{\mathcal{A}} n_{\mathcal{B}})$ for computing the description of the boundary of $\mathcal{CB}$, when both $\mathcal{A}$ and $\mathcal{B}$ are nonconvex. In the case where $\mathcal{A}$ is nonconvex, using this algorithm first and projecting the C obstacles on the x–y plane next would probably be an efficient method. However, we have not implemented it.

F. Swept-Area Method

1) Principle: The swept-area method consists of first computing two areas swept out by $\mathcal{A}$ when it rotates about its reference point from orientation γ_u to orientation γ_{u+1}: the outer swept area denoted by $\mathcal{OSA}[\gamma_u, \gamma_{u+1}]$ and the inner swept area denoted by $\mathcal{ISA}[\gamma_u, \gamma_{u+1}]$. They are defined as follows:[5]

$$\mathcal{OSA}[\gamma_u, \gamma_{u+1}] = \bigcup_{\theta \in [\gamma_u, \gamma_{u+1}]} \mathcal{A}(0,0,\theta) = \{(x,y)/\exists \theta \in [\gamma_u, \gamma_{u+1}] : (x,y) \in \mathcal{A}(0,0,\theta)\}$$

$$\mathcal{ISA}[\gamma_u, \gamma_{u+1}] = \bigcap_{\theta \in [\gamma_u, \gamma_{u+1}]} \mathcal{A}(0,0,\theta) = \{(x,y)/\forall \theta \in [\gamma_u, \gamma_{u+1}] : (x,y) \in \mathcal{A}(0,0,\theta)\}.$$

Then the method treats both $\mathcal{OSA}[\gamma_u, \gamma_{u+1}]$ and $\mathcal{ISA}[\gamma_u, \gamma_{u+1}]$ as moving objects which can only translate in the plane, and it maps the obstacle $\mathcal{B}$ in the configuration spaces (copies of $\boldsymbol{R}^2$) of these objects. The mapping yields two regions: $\mathcal{OB}[\gamma_u, \gamma_{u+1}]$ and $\mathcal{IB}[\gamma_u, \gamma_{u+1}]$, which are formally

[5] The outer swept area is similar to that described in [21].

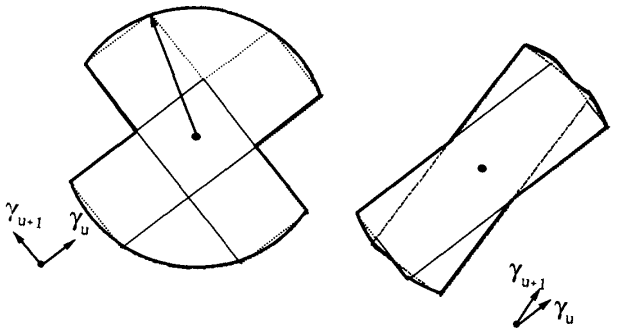


Fig. 7. Outer swept areas.

defined as follows see (Section II-B):

$$\mathcal{OB}[\gamma_u, \gamma_{u+1}] = \mathcal{B} \ominus \mathcal{OSA}[\gamma_u, \gamma_{u+1}] \quad \text{and}$$
$$\mathcal{IB}[\gamma_u, \gamma_{u+1}] = \mathcal{B} \ominus \mathcal{ISA}[\gamma_u, \gamma_{u+1}].$$

The Minkowski difference of two generalized polygons is a generalized polygon [20], which can be computed in time $O(n_1 + n_2)$ if the two input polygons are convex, and time $O(n_1^2 n_2^2 \log n_1 n_2)$ otherwise. (n_1 and n_2 denote the number of edges of the input generalized polygons.)

The swept-area method returns two regions:

$$[x_1, x_2] \times [y_1, y_2] \cap \mathcal{OB}[\gamma_u, \gamma_{u+1}] \quad \text{and}$$
$$[x_1, x_2] \times [y_1, y_2] \cap \mathcal{IB}[\gamma_u, \gamma_{u+1}].$$

The first is exactly $\mathcal{OCB}_{xy}[\kappa^u]$, as is shown in [21]. The second is strictly included in $\mathcal{ICB}_{xy}[\kappa^u]$, which leads the overall decomposition algorithm to generate a set of FULL cells of less total volume than with the projection method.

In practice, the swept-area method is significantly more efficient than the projection method when the intervals $[\gamma_u, \gamma_{u+1}]$ are large. When these intervals become small enough, the projection method is preferred because it exactly computes $\mathcal{ICB}_{xy}[\kappa^u]$.

2) Computation of $\mathcal{OSA}[\gamma_u, \gamma_{u+1}]$: The outer swept area $\mathcal{OSA}[\gamma_u, \gamma_{u+1}]$ is a generalized polygon (see Fig. 7). The straight edges of this polygon are portions of edges of $\mathcal{A}(0, 0, \gamma_u)$ and $\mathcal{A}(0, 0, \gamma_{u+1})$. Each circular edge is the locus of a vertex of $\mathcal{A}$ when $\mathcal{A}$ rotates from γ_u to γ_{u+1} about the reference point.

3) Computation of $\mathcal{ISA}[\gamma_u, \gamma_{u+1}]$: The inner swept area $\mathcal{ISA}[\gamma_u, \gamma_{u+1}]$ is a generalized polygon (see Fig. 8). The straight edges of this polygon are portions of the edges of $\mathcal{A}(0, 0, \gamma_u)$ and $\mathcal{A}(0, 0, \gamma_{u+1})$. Each circular edge is the locus of a point obtained by projecting the reference point on an edge of $\mathcal{A}$ if that projection falls between the two endpoints of the edge. The contour of $\mathcal{ISA}[\gamma_u, \gamma_{u+1}]$ is traced out by starting at the first intersection of a ray (any one) drawn from the reference point $\mathcal{O}_{\mathcal{A}}$ with the potential edges of $\mathcal{ISA}[\gamma_u, \gamma_{u+1}]$. (Since $\mathcal{O}_{\mathcal{A}}$ is in the interior of $\mathcal{A}$, it is also in the interior of $\mathcal{ISA}[\gamma_u, \gamma_{u+1}]$.)

IV. Hierarchical Graph Searching

A. Improved Algorithm

The major drawback of the simple search algorithm outlined in Section II-D is that the search work performed in the connectivity graph G_i, if it does not return success, is not used to help the search of G_{i+1}. This drawback can be remedied as follows.

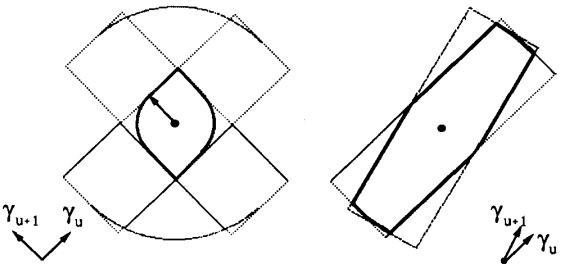


Fig. 8. Inner swept areas.

Rather than reconstructing and searching the complete connectivity graph (CG) when the MIXED cells along an M channel are refined, a separate CG representing the decomposition of every refined cell κ is generated and recursively searched for a "subchannel." This subchannel is a sequence of adjacent EMPTY or MIXED cells produced by the decomposition of κ.

The new algorithm hence generates a hierarchy of CG's. The CG at the top of the hierarchy corresponds to the initial decomposition of $\mathcal{C}$—i.e., $\mathcal{H}$—and is denoted by $G^{\mathcal{C}}$. Every other CG corresponds to the decomposition of a certain MIXED cell, say κ, and is denoted by G^{κ}. A channel Π_1 is first generated in $G^{\mathcal{C}}$. If Π_1 is an E channel, the planner exits with success; otherwise, each MIXED cell κ in Π_1 is decomposed recursively, and a subchannel Π^{κ}, if any, is generated in G^{κ}. This subchannel is substituted for κ in Π_1.

In order to make the algorithm work properly, one must be careful that each subchannel Π^{κ} connects appropriately to the rest of Π_1 [13]. This can be worked out as explained below by generating a complete channel Π_2 connecting q_{init} to q_{goal}. Π_2 is first set to the empty sequence of cells and is then incrementally augmented into a refinement of Π_1. Each cell κ in Π_1 is considered in the order it appears in Π_1. If κ is EMPTY, it is simply appended to the current Π_2. If κ is MIXED, it is first decomposed into a set $\mathcal{P}^{\kappa}$ of smaller cells and a CG G^{κ} is built using this decomposition; then, G^{κ} is searched for a subchannel Π^{κ} satisfying the following four conditions:

- If κ is the first cell in Π_1 (hence, it contains q_{init}), then the first cell in Π^{κ} also contains q_{init}.
- If κ is the last cell in Π_1 (hence, it contains q_{goal}), then the last cell in Π^{κ} also contains q_{goal}.
- If κ is not the first cell in Π_1, the first cell of Π^{κ} is adjacent to the last cell of the current Π_2.
- If κ is not the last cell in Π_1, the last cell of Π^{κ} is adjacent to the cell following κ in Π_1.

In the following, a sequence of adjacent EMPTY or MIXED cells in the decomposition of κ is called a subchannel iff it satisfies these conditions.

Assuming that the planner succeeds in generating Π^{κ}, Π_2 is modified by appending Π^{κ} to it. The case when the planner fails in generating Π^{κ} will be examined in Section IV-C.

When all the cells in Π_1 have been considered successfully, we have obtained a channel Π_2, which is a refinement of Π_1. If Π_2 is an E channel, the planning problem is solved. Otherwise, Π_2 is recursively refined into a third channel Π_3 by decomposing the MIXED cells it contains, etc.

B. Cell Occurrences in a Channel

An E channel is generated by refining an M channel without reproducing and searching a global CG. Therefore, it is impor-

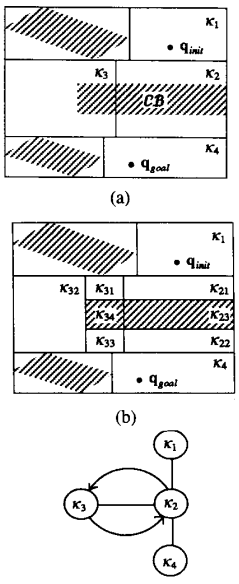


Fig. 9. Multiple cell occurrences in an M channel.

tant to allow an M channel to contain several occurrences the same MIXED cell. This is best explained using an example.

Consider the two-dimensional example of Fig. 9. In Fig. 9(a), the sequence of cells $(\kappa_1, \kappa_2, \kappa_3, \kappa_2, \kappa_4)$, which contains κ_2 twice, has to be considered as an M channel. (The grey area represents C obstacles so that κ_2 and κ_3 are MIXED, whereas κ_1 and κ_4 are EMPTY.) We assume that κ_2 and κ_3 are next decomposed as shown in Fig. 9(b). When the two occurrences of κ_2 are refined, they produce different EMPTY subchannels. This is illustrated in Fig. 9(b), where κ_2 is partitioned into three smaller cells: two EMPTY ones κ_{21} and κ_{22} and one FULL cell κ_{23}. The first occurrence of κ_2 leads to a one-cell subchannel made of κ_{21}, whereas the second occurrence leads to another one-cell subchannel made of κ_{22}. On the other hand, the decomposition of κ_3 produces four cells: three EMPTY ones κ_{31}, κ_{32}, and κ_{33} and a FULL one κ_{34}. From this decomposition the EMPTY subchannel $(\kappa_{31}, \kappa_{32}, \kappa_{33})$ is extracted. The generated E channel is $(\kappa_1, \kappa_{21}, \kappa_{31}, \kappa_{32}, \kappa_{33}, \kappa_{22}, \kappa_4)$.

Since the same cell may appear several times in a channel Π_i, from here on, we refer to the elements in a channel as cell occurrences rather than just cells. We denote by ω_i^k the kth cell occurrence in Π_i and cell(ω), the cell of which ω is an occurrence. A cell occurrence ω is said MIXED (resp. EMPTY) iff cell(ω) is MIXED (resp. EMPTY). A subchannel generated as a refinement of a MIXED cell occurrence ω is denoted by Π^{ω}; it is constructed by searching $G^{\text{cell}(\omega)}$.

C. Failure Recovery

Consider now the situation where the planner is constructing the channel Π_{i+1} and fails to refine a MIXED cell occurrence ω_i^j contained in Π_i into a subchannel. We have explicitly mentioned this situation in Section IV-A. It may be caused either by the fact that Π_i does not contain an E channel as illustrated in Fig. 10 or by the fact that the partial channel Π_{i+1} generated so far is a blind alley, as is illustrated in Fig. 11.

Fig. 10(a) shows an M channel $\Pi_1 = (\kappa_1, \kappa_2, \kappa_3)$ in bold lines. Assume that κ_1 and κ_2 are decomposed as shown in Fig.

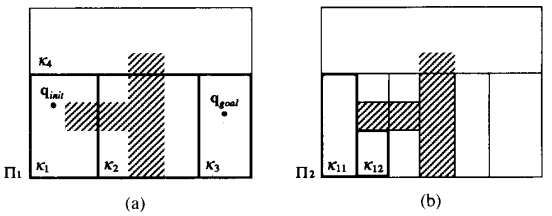


Fig. 10. First type of failure.

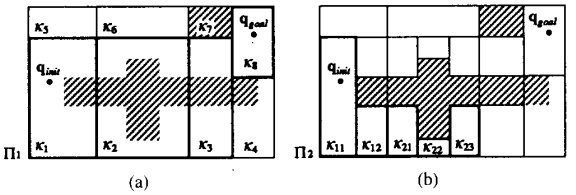


Fig. 11. Second type of failure.

10(b). The subchannel $(\kappa_{11}, \kappa_{12})$ is constructed within κ_1, but then, the planner fails to find a subchannel within κ_2 because a C obstacle obstructs the passage between κ_1 and κ_3 through κ_2. Recovering from this failure requires the planner to generate an alternative channel Π_1'.

Fig. 11(a) shows an M channel $\Pi_1 = (\kappa_1, \kappa_2, \kappa_3, \kappa_8)$. Assume that κ_1, κ_2 and κ_3 are decomposed as shown in Fig. 11(b) and that the incomplete channel $\Pi_2 = (\kappa_{11}, \kappa_{12}, \kappa_{21}, \kappa_{22}, \kappa_{23})$ has been generated. Then, the planner fails to find a subchannel in the decomposition of κ_3. Recovering from this failure requires the planner to generate alternative subchannels in κ_1 and κ_2. Of course, the planner has no way to directly determine the cause of the failure. Instead, when it fails to refine the MIXED cell occurrence ω_i^j into a subchannel, it distinguishes among the three cases described below, which are mutually exclusive:

Case (1): $j = 1$: $\omega_i^j = \omega_i^1$ is the first cell occurrence in Π_i. The failure typically means that there is no way to connect q_{init} to the second cell occurrence in Π_i. The only alternative for the planner is to attempt to generate a new channel Π_i. If no new channel Π_i can be generated and $i > 1$, the planner recursively attempts to generate a new channel Π_{i-1}. If no new channel Π_i can be generated and $i = 1$, the planner returns failure.

Case (2): $j > 1$ and ω_i^{j-1} is EMPTY: The failure typically means that there is no way to connect any configuration in the EMPTY cell occurrence ω_i^{j-1} to a configuration in ω_i^{j+1} through ω_i^j. The planner then proceeds as described in **Case (1)**. (Some differences between these two cases will be made apparent in the next subsection.)

Case (3): $j > 1$ and ω_i^{j-1} is MIXED: The failure typically means that there is no way to connect any configuration in the last cell occurrence of the current Π_{i+1} (which is a subset of ω_i^{j-1}) to a configuration in ω_i^{j+1} through ω_i^j. The planner attempts to generate another subchannel $\Pi^{\omega_i^{j-1}}$ in the CG of cell(ω_i^{j-1}). If it finds one, the planner substitutes it for the previous one in Π_{i+1} and tries again to refine ω_i^j into a subchannel by searching the CG of cell(ω_i^j). Otherwise, it iteratively treats ω_i^{j-1} as it just treated ω_i^j, i.e., it sets j to $j - 1$ and applies the treatment of **Cases (1)**, **(2)**, or **(3)**, whichever is applicable.

D. Recording Failure Conditions

Within the framework described above, the same CG may be searched multiple times for (sub)channels. The planner can be

significantly more efficient if it remembers the "mistakes" it has made by annotating the relevant cells. These annotations can be used by the planner to avoid recommitting the same mistakes. Cell annotation is performed whenever the planner recovers from the failure to refine a cell occurrence.

Constructing and remembering all failure conditions can be quite involved and costly. For these reasons, we restrict the generation of annotations to **Cases (1)** and **(2)** presented in the previous subsection when these cases do not occur after an iteration on **Case (3)**. The resulting annotations are limited in number, and they are very simple to use. (The symbol n_i denotes the number of cell occurrences in Π_i. The symbol "λ" stands for the inexistent cell.)

Case (1): Let $\xi = \text{cell}(\omega_i^1)$ and $\psi = \text{cell}(\omega_i^2)$ be the first two cells in the current Π_i ($\psi = \lambda$ if $n_i = 1$). The cell ξ is annotated with (ψ).

This annotation is later used as follows: If a new channel Π_i is generated and if it contains an occurrence ω_ξ of ξ in first position, then an occurrence of ψ should not be considered a valid successor of ω_ξ.

Notice that the first cell occurrence in the new Π_i is necessarily an occurrence of ξ since ξ contains q_{init}, but there may be other occurrences of ξ appearing in this new Π_i to which the above annotation does not apply.

Case (2): Let $\xi = \text{cell}(\omega_i^j)$ ($j > 1$), $\varphi = \text{cell}(\omega_i^{j-1})$ and $\psi = \text{cell}(\omega_i^{j+1})$ in the current Π_i. If $j = n_i$, then $\psi = \lambda$. ξ is annotated with (φ, ψ).

This annotation is later used as follows: If a new channel Π_i is generated and if it contains an occurrence ω_ξ of ξ preceded by an occurrence of φ, then an occurrence of ψ should not be considered a valid successor of ω_ξ. In the particular case where $\psi = \lambda$, this means that ω_ξ must not be the last cell occurrence of the new Π_i if it is preceded by an occurrence of φ.

Notice that if $\psi \neq \lambda$, the above annotation is commutative, i.e., if ξ cannot be traversed by a subchannel connecting φ to ψ, it can neither be traversed by a subchannel connecting ψ to φ. Therefore, the annotation (φ, ψ) is used commutatively whenever $\psi \neq \lambda$.

E. Search of a Connectivity Graph

Below, we regard $\mathcal{X}$ as a cell, and we denote the channel (ω_0^1) with $\text{cell}(\omega_0^1) = \mathcal{X}$ by Π_0. (We assume that Π_0 is an M channel.) The channel Π_1 is a refinement of Π_0.

Assume that the planner has already generated an M channel Π_i ($i \geq 0$), which it is now refining into Π_{i+1} and that it is currently considering the MIXED cell occurrence $\omega_i^j \in \Pi_i$. Let $\kappa = \text{cell}(\omega_i^j)$. If κ has already been decomposed, the planner just reuses the CG G^κ that was previously generated; otherwise, it decomposes κ into a set $\mathcal{P}^\kappa$ of smaller rectangloid cells and builds G^κ. It then searches G^κ for a subchannel refining ω_i^j that will be appended to the current Π_{i+1}.

The search of G^κ first requires the possible initial and goal cells to be determined:

- If $j = 1$, ω_i^j is the first cell of Π_i. Then, the only possible initial cell in G^κ is the cell of $\mathcal{P}^\kappa$ that contains q_{init}. Otherwise, the possible initial cells of G^κ are all the cells of $\mathcal{P}^\kappa$ that are adjacent to the cell—call it φ—of the last occurrence ω_φ in the current Π_{i+1} and whose insertion in Π_{i+1} as cell occurrences succeeding ω_φ does not violate any annotation.
- If $j = n_i$, ω_i^j is the last cell occurrence of Π_i. The only possible goal cell in G^κ is the cell of $\mathcal{P}^\kappa$ that contains q_{goal}. Otherwise, the possible goal cells of G^κ are all the cells of $\mathcal{P}^\kappa$ that are adjacent to $\text{cell}(\omega_i^{j+1})$.

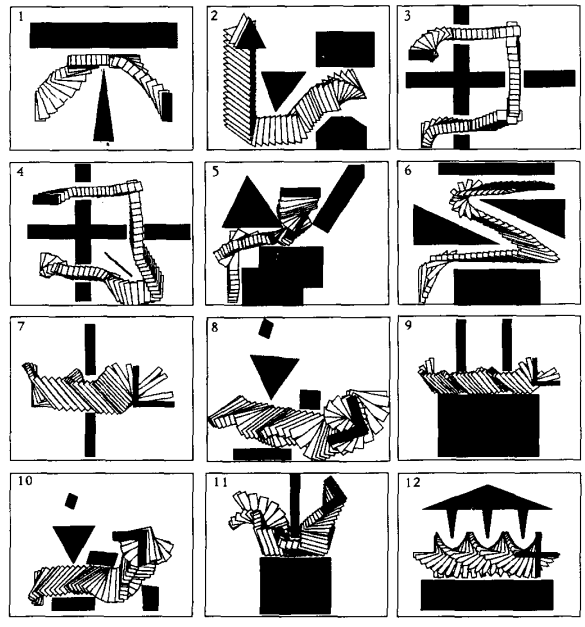


Fig. 12. Examples.

If there is no possible initial or goal cell in G^κ, the planner considers this situation as a failure to refine ω_i^j and applies the treatments described in Sections IV-C and D. If at least one initial and one goal cell are established in G^κ, the planner searches G^κ for a subchannel $\Pi^{\omega_i^j}$, linking any of the possible initial cells to any of the possible goal cells. It does that in such a way that no annotation is violated.

The subchannel $\Pi^{\omega_i^j}$ to be generated in G^κ can have arbitrarily many occurrences of a MIXED cell in $\mathcal{P}^\kappa$. This means that there may be an infinity of potential subchannels to consider. One way to deal with this difficulty is by limiting the number of occurrences of the same cell in a subchannel. This limitation seems to be reasonable in most practical cases.

V. Implementation and Experimentation

We have implemented the algorithms described in the above sections in a path planner. The planner is written in Allegro Common Lisp and runs on an Apple Macintosh II computer.

We ran the planner on many examples including both convex and nonconvex moving objects. Twelve of these examples are illustrated in Fig. 12. In these figures, we show a path extracted from the generated channel by connecting the center points of the entrance and exit of each cell along the channel by a straight line segment. Statistics—CPU time, total number of cells generated (N_{total}), number of EMPTY cells generated (N_{EMPTY}), and number of EMPTY cells used in the output channel (N_{used})—characterizing the efficiency of the planner are given in Table I. The CPU time covers all the processing steps of the planner between the input of the problem by the user to the output of the path.

Examples 5 and 8 have been previously reported in [4] (BLP planner), which is the best previous planner known to the authors using the hierarchical approximate cell decomposition approach. In Table II, we give the comparison of the statistics

TABLE I
STATISTICS FOR THE TWELVE EXAMPLES

example	CPU time (min)	N_{total}	N_{EMPTY}	N_{used}
1	0.6	98	35	12
2	0.9	140	74	18
3	1.5	210	104	10
4	2.5	264	134	28
5	5.5	293	96	36
6	5.0	218	116	29
7	0.9	160	77	22
8	2.5	205	88	17
9	6.5	170	25	13
10	9.8	206	46	19
11	11.0	312	72	21
12	10.5	369	121	18

TABLE II
COMPARISON WITH THE BLP PLANNER

	Example 5				Example 8			
planner	CPU time	N_{total}	N_{EMPTY}	N_{used}	CPU time	N_{total}	N_{EMPTY}	N_{used}
Ours	5.5 mins	293	96	36	2.5 mins	205	88	17
BLP	tens of mins	644	120	87	tens of mins	782	62	29

on these two examples with our planner and BLP planner.[6] We should point out that for Example 5 the path generated by our planner is simpler than the path generated by BLP planner, i.e., it does not include a backing-up maneuver.

For examples 2, 6, 11, and 12, we compared the number of cells generated by our planner with the number of cells that a similar planner using the octree decomposition technique (see Section III-B) would generate. Based on partial implementation, we obtained a conservative estimate (i.e., lower bound) of the number of cells generated by the octree decomposition technique. The data are reported in Table III. Example 11 was extracted from [2]. According to this paper, the planner described in [10], which uses the octree decomposition technique, generates more than 10 000 cells. Not only our planner generates less cells, but the number of generated cells grows less quickly when the difficulty of the problem increases.

The above results demonstrate major improvements brought by our algorithms to the approximate cell decomposition approach.

In addition to the previous experiments, we tried to characterize empirically the efficiency of the planner as a function of the "complexity" of the workspace, i.e., the number of edges of the obstacles and the "sparsity" of the workspace, i.e., the distance between obstacles. To that purpose, we have run our planner on several hundred problems using randomly generated environments with prespecified complexity and sparsity. Fig. 13 shows twelve of these examples.[7] For each complexity and sparsity, we have generated and run tens of examples. The mean value of the running time obtained for some series of examples is depicted in Fig. 14. Fig. 14(a) corresponds to the first row of Fig. 13, Fig. 14(b) to the third row, Fig. 14(c) to the first column, and Fig. 14(d) to the third column. Figs. 14(a) and (b) show that within the range of our experiments, the running time varies linearly with the complexity. Figs. 14(c) and (d) show

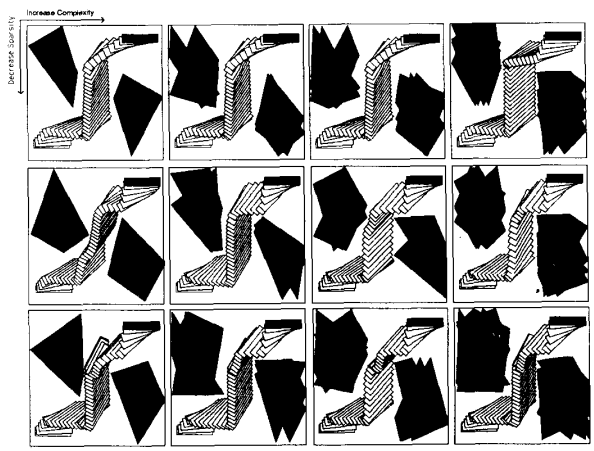


Fig. 13. Randomly generated examples.

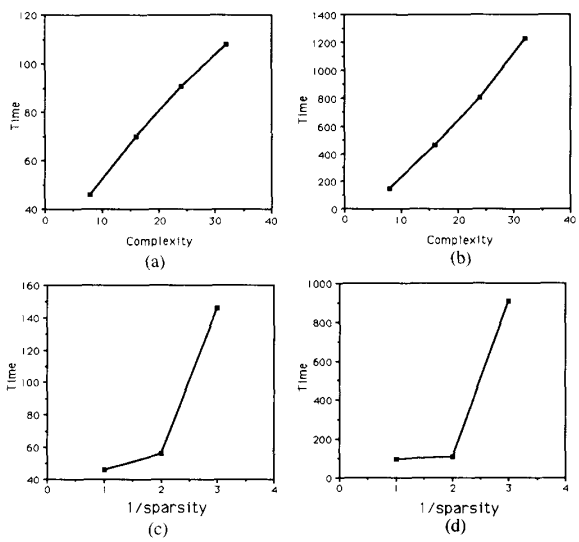


Fig. 14. Statistics for the randomly generated examples.

[6] The BLP planner was implemented on a LISP machine, which runs LISP approximately five times faster than the Macintosh II runs Allegro Common Lisp (according to our own benchmark). In addition, since we do not count the number of FULL cells in our planner's statistics (they have no effect on the search graph), we subtracted this number from the statistics of the BLP planner.

[7] In the example of Fig. 13, the two obstacles in the workspace are enclosed within two rectangular boxes. The sparsity is measured as the distance between the vertical sides of these two boxes. When there are more than two obstacles, the sparsity can be defined as the minimum distance between two boxes.

TABLE III
COMPARISON WITH THE OCTREE METHOD

Num cells	*Example2*	*Example6*	*Example11*	*example12*
Ours	140	218	312	369
Octree	> 500	> 2000	> 5000	> 5000

that above some clutteredness (inverse of sparsity), the running time increases quickly. This corresponds to the fact that the planner has to generate many small cells before it can find an E channel.

VI. CONCLUSION

We have described new heuristic algorithms for path planning using the hierarchical approximate cell decomposition approach. An important feature of this approach is that it decomposes the robot's configuration space into cells of predefined shape—typically, rectangloids—at successive levels of approximation. The two main steps of the approach are cell decomposition and graph searching.

With respect to cell decomposition, we have proposed a "constraint reformulation" approach, which attempts to minimize the total volume of the generated MIXED cells while producing a reasonably small number of cells. We have described specific algorithms for implementing this approach. These algorithms construct bounding and bounded rectangloid approximations of the C obstacles. Experiments show that the constraint reformulation approach with these algorithm produces much better results than earlier decomposition techniques.

With respect to graph searching, we have proposed a set of techniques for allowing the planner to take advantage of the work already carried out at previous levels of detail. These techniques, inspired by dependency-directed backtracking systems, allows the planner to record and use the conditions of past mistakes so that it does not repeat them.

We have implemented these algorithms in a path planner with which we have conducted a variety of experiments. These experiments show that our planner is significantly faster than previous planners based on the same general approach.

Our current research is aimed at developing a motion planner/controller capable of dealing with some unexpected obstacles. Our approach is described in [8]. It consists of controlling the robot in a channel using a potential field approach. We have conducted experiments with this approach both in a computer-simulated environment and using an actual robot. We currently work on the interaction between the planner and the controller so that if the robot gets stuck in a channel (for example, the channel may be completely obstructed by an unexpected obstacle), the controller may ask the planner to modify the channel accordingly. This kind of interaction requires the planner to be sufficiently fast.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*. Reading, MA: Addison-Wesley, 1983.

[2] F. Avnaim and J. D. Boissonnat, "Polygon placement under translation and rotation," INRIA, Sophia-Antipolis, France, Tech. Rep. 889, 1988.

[3] D. Ayala, P. Brunet, R. Juan, and I. Navazo, "Object representation by means of nonminimal division quadtrees and octrees," *ACM Tran. Graphics*, vol. 4, no. 1, 1985.

[4] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," AI Lab., MIT, Cambridge, MA, AI Memo 684, 1982.

[5] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," in *Proc. 8th Int. Joint Conf. Artificial Intell.* (Karlsruhe, FRG), 1983, pp. 799-806.

[6] R. C. Brost, "Computing metric and topological properties of configuration-space obstacles, in *Proc. IEEE Int. Conf. Robotics Automat.* (Scottsdale, AZ), 1989, pp. 170-176.

[7] I. Carlbom *et al.*, "A hierarchical data structure for representing the spatial decomposition of 3D objects," in *Computer Graphics: Theory and Applications*, T. L. Kunii, Ed. New York: Springer-Verlag, 1983, pp. 96-110.

[8] W. Choi, D. J. Zhu, and J. C. Latombe, "Contingency-tolerant motion planning and control," in *Proc. IEEE/RSJ Int. Workshop Intell. Robots Syst., IROS'89* (Tsukuba, Japan), 1989, pp. 78-85.

[9] B. R. Donald, "Motion planning with six degrees of freedom," Artificial Intell. Lab., MIT, Cambridge, MA, Rep. AI-TR-791, 1984.

[10] B. Faverjon, "Object level programming of industrial robots," in *Proc. IEEE Int. Conf. Robotics Automat.* (San Francisco, CA), 1986, pp. 1406-1412.

[11] K. Fujimura *et al.*, "Oct-tree algorithms for solid modeling," in *Frontiers Comput. Graphics: Proc. Comput. Graphics Tokyo '84*, 1984, pp. 2-12.

[12] L. Gouzènes, "Strategies for solving collision-free trajectories problems for mobile and manipulator robot," in *Proc. Int. J. Robotics Res.*, vol. 3, no. 4, pp. 51-65, 1984.

[13] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE J. Robotics Automat.*, vol. RA-2, no. 3, pp. 135-145, 1986.

[14] K. Kant and S. W. Zucker, "Planning smooth collision-free trajectories: Path, velocity and splines in free space," Comput. Vision Robotics Lab., McGill University, Montréal, Tech. Rep., 1986.

[15] K. Kedem and M. Sharir, "An efficient motion planning algorithm for a convex polygonal object in 2-dimensional polygonal space," Courant Inst. Math. Sci., New York Univ., New York, NY, Tech. Rep. 253, 1986.

[16] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robotics Res.*, vol. 5, no. 1, pp. 90-98, 1986.

[17] P. Khosla and R. Volpe, "Superquadric artificial potentials for obstacle avoidance and approach," in *Proc. IEEE Int. Conf. Robotics Automat.* (Philadelphia, PA), 1988, pp. 1178-1784.

[18] J. C. Latombe, "Failure processing in a system for designing complex assemblies," in *Proc. 6th Int. Joint Conf. Artificial Intell.* (Tokyo, Japan), 1979.

[19] C. Laugier and F. Germain, "An adaptative collision-free trajectory planner," *Int. Conf. Advanced Robotics* (Tokyo, Japan), 1985.

[20] J. P. Laumond, "Obstacle growing in a non polygonal world," *Inform. Processing Lett.*, vol. 25, pp. 41-50, 1987.

[21] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 108-120, 1983.

[22] ——, "A simple motion-planning algorithm for general robot manipulators," *IEEE J. Robotics Automat.*, vol. RA-3, no. 3, pp. 224-238, 1987.

[23] C. O'Dúnlaing, M. Sharir, and C. K. Yap, "Retraction: A new approach to motion planning," in *Proc. 15 FOCS*, 1983, pp. 207-220.

[24] J. Ponce and D. Chelberg, "Localized intersections computation for solid modelling with straight homogeneous generalized cylinders," in *Proc. DARPA Image Understanding Workshop*.

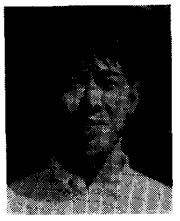
[25] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.

[26] E. Rimon and D. E. Koditschek, "Exact robot navigation using cost functions: The case of distinct spherical boundaries in E^n," in *Proc. IEEE Int. Conf. Robotics Automat.* (Philadelphia, PA), 1988, pp. 1791-1796.

[27] J. T. Schwartz and M. Sharir, "On the 'piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds," in *Advances in Applied Mathematics*. New York: Academic.

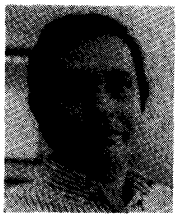
[28] J. T. Schwartz, M. Sharir, and J. Hopcroft, *Planning, Geometry, and Complexity of Robot Motion*. Norwood, NJ: Ablex, 1987.

[29] Spivak, M., *A Comprehensive Introduction to Differential Geometry*. Wilmington, DE: Publish or Perish, 1979.
[30] R. M. Stallman and G. J. Sussman, "Forward reasoning and dependency-directed backtracking in a system for computed-aided circuit analysis," *Artificial Intell.*, vol. 9, no. 2, pp. 135–196.



David J. Zhu received the B.A. degree in mathematics and physics from Franklin and Marshall College, Lancaster, PA, in 1986. He is currently completing the Ph.D. degree at the Robotics Laboratory in Computer Science Department at Stanford University, Stanford, CA.

His research interests include AI, spatial reasoning, motion planning, and automatic pipe layout design.



Jean-Claude Latombe received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in computer science from the National Polytechnic Institute of Grenoble, France, in 1969, 1972, and 1977, respectively.

From 1980 to 1984, he was a faculty member at Ecole Nationale Superieure d'Informatique et de Mathematiques Appliquees de Grenoble (ENSIMAG), where he developed new technical programs in artificial intelligence, computer vision, and robotics. From 1984 to 1987, he was on leave from ENSIMAG and held the position of the executive president of Industry and Technology for Machine Intelligence (ITMI), a company that he cofounded in 1982 for commercializing robotic systems and expert systems. From 1984 to 1987, he directed the French national project in Artificial Intelligence supported by the Ministry of Research. In April 1987, he joined Stanford University as Associate Professor in the Department of Computer Science. At Stanford, he is the Director of the Computer Science Robotics Laboratory.