Solving the Find-Path Problem by Good Representation of Free Space

RODNEY A. BROOKS

Abstract—**Free space is represented as a union of (possibly overlapping) generalized cones. An algorithm is presented which efficiently finds good collision-free paths for convex polygonal bodies through space littered with obstacle polygons. The paths are good in the sense that the distance of closest approach to an obstacle over the path is usually far from minimal over the class of topologically equivalent collision-free paths. The algorithm is based on characterizing the volume swept by a body as it is translated and rotated as a generalized cone, and determining under what conditions one generalized cone is a subset of another.**

I. Introduction

The FIND-PATH problem is well-known in robotics. Given an *object* with an initial location and orientation, a goal location and orientation, and a set of *obstacles* located in space, the problem is to find a continuous path for the object from the initial position to the goal position which avoids collisions with obstacles along the way. This paper presents a new representation for free space as natural "freeways" between obstacles, and a new algorithm to solve find-path using that representation. The algorithm's advantages over those previously presented are that it is quite fast and it finds good paths which generously avoid obstacles rather than barely avoiding them. It does not find possible paths in extremely cluttered situations, but it can be used to provide direction to more computationally expensive algorithms in those cases.

A. Approaches to the Find-Path Problem

A common approach to the find-path problem, used with varying degrees of sophistication, is the configuration space approach. Lozano-Pérez [5] gives a thorough mathematical treatment of configuration space. The idea is to determine those parts of free space which a reference point of the moving object can occupy without colliding with any obstacles. A path is then found for the reference point through this truly free space. Dealing with rotations turns out to be a major difficulty with the approach, requiring complex geometric algorithms which are computationally expensive.

Manuscript received June 11, 1982; revised September 28, 1982. This paper was originally presented at and appeared in the Proceedings of the Second National Conference of the American Association for Artificial Intelligence, Pittsburgh, PA, August 1981. This work was supported in part by the Office of Naval Research under Contract N00014-81-K-0494, and in part by the Advanced Research Projects Agency under Office of Naval Research Contract N00014-80-C-0505.

The author is with The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139.

Conceptually one can view configuration space as shrinking the object to a point while at the same time expanding the obstacles to the shape of the moving object. This approach works well when the moving object is not allowed to rotate. If it can rotate, then the grown obstacles must be embedded in a higher dimensional space—one extra dimension for each degree of rotational freedom. Furthermore the grown obstacles have nonplanar surfaces, even when the original problem was completely polygonal or polyhedral. Typically implementors have approximated the grown obstacles in order to be able to deal with rotations.

Moravec [6] had to solve the find-path problem in two dimensions. He bounded all obstacles and the moving object by circles. Then the grown obstacles were all perpendicular cylinders and the problem could be projected back into two dimensions where rotations could be ignored. This method missed all paths that required rotational maneuvering. Lozano-Pérez [5] split the rotation range into a fixed number of slices, and within each slice bounded the ground obstacles by polyhedra. Earlier Udupa [8] had used much cruder bounding polyhedra in a similar way.

Recently Brooks and Lozano-Pérez [3] developed a method for computing more directly with the curved surfaces of the grown obstacles. The algorithm can be quite expensive although given enough time and space it can find a path if one exists.

B. Good Representations Simplify Find-Path

The algorithm presented in this paper is based on a different idea, i.e., that of using representations of space which capture the essential effects of translating and rotating a body through space. Free space is represented as overlapping generalized cones since they are descriptions of swept volumes. The cones provide a high-level plan for moving the object through space. The volume swept by the particular object as it is translated and rotated is characterized as a function of its orientation. Find-path then reduces to comparing the swept volume of the object with the sweepable volumes of free space. This is done by inverting the characterization of the volume swept by an object, to determine its valid orientation as it moves through a generalized cone. Throughout this paper we restrict our attention to the two-dimensional problem where the object to be moved is a convex polygon and the obstacles are represented as unions of convex polygons.

II. Describing Free Space as Generalized Cones

Generalized cones are a commonly used representation of volume for modeling objects in artificial intelligence-based computer vision systems. They were first introduced by Binford [2]. A generalized cone is formed by sweeping a two-dimensional *cross section* along a curve in space, called a *spine*, and deforming it according to a *sweeping rule*.

We will consider a two-dimensional specialization of generalized cones (although we will still refer to them as volumes). The spines will be straight and the cross sections will be line segments held perpendicular to the spine with left and right radii. The sweeping rule will independently control the magnitudes of the left and right radii as piecewise linear functions.

Free space is represented as overlapping generalized cones. Natural "freeways," elongated regions through which the object might be moved, are represented as generalized cones. The generalized cones overlap at intersections of these natural freeways. Fig. 1 illustrates a few of the generalized cones describing the free space around three obstacles in a confined workspace.

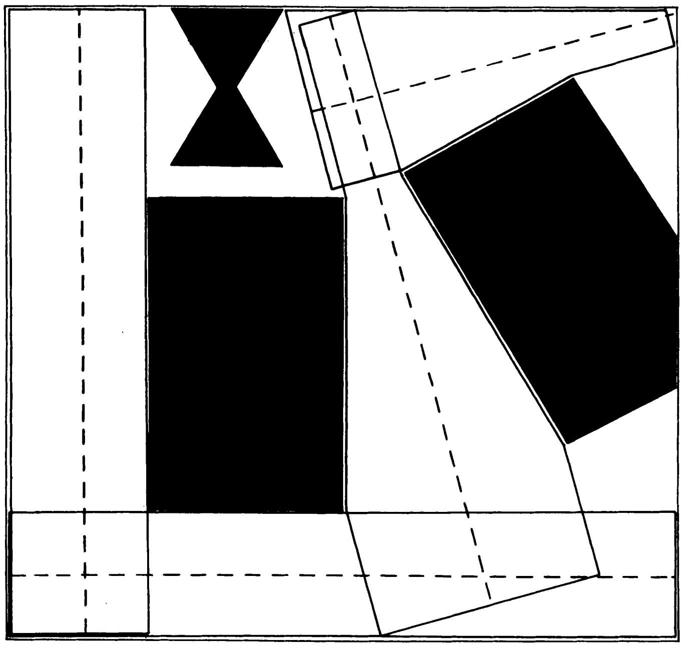
Fig. 1. Generalized cones generated by two obstacles and workspace boundary.

A. Candidate Generalized Cones

A representation of free space is constructed by examining all pairs of edges of obstacle polygons. If the edges define a natural "freeway" through space they are used to construct a generalized cone. Each edge has a "free" side and a "full" side, where the "free" side is the outside of the particular polygon with which it is associated. Each edge has an outward pointing normal, pointing into the "free" side. Two edges are accepted as defining a *candidate generalized cone* if they meet the following requirements. 1) At least one vertex of each edge should be on the "free" side of the other, and 2) the dot product of the outward pointing normals should be negative. The second condition ensures that the "free" sides of the two edges essentially face each other. Thus in this initial stage there is a complexity factor of $O(n^2)$, where n is the number of edges of obstacle polygons.

Given a candidate pair of edges, a spine is constructed for the generalized cone. It is the bisector of the space which is on the "free" side of both edges. (Thus if the edges are parallel the spine is parallel and equidistant to them both, or if the edges are not parallel, then the spine bisects the angle they form.) The generalized cone occupies the volume between the two defining edges. At each vertex of the two edges (if the vertex is on the "free" side of the other edge) the cone is extended parallel to the spine.

The generalized cone so defined may not lie entirely in free space. There may be other obstacles which intersect it. Each obstacle is compared to the generalized cone. A polygon can be intersected with the cone in time $O(n)$, where n is the number of edges of the polygon. If the intersection is empty, then nothing further need be done. If not, then the intersection is projected normally onto the spine of the generalized cone. This is illustrated in Fig. 2. Again this is an $O(n)$ operation in the number of vertices (and hence edges). The result of comparing all obstacles to the generalized cone is a set of regions of the spine where there is no obstacle which intersects the cone in a slice normal to the spine. Each disjoint slice which includes parts of the original two edges is then accepted as a generalized cone describing part of free space. Clearly the complete operation of describing free space is at most $O(n^3)$ in the number of edges in the obstacle polygons.

B. Representation Used for Generalized Cones

Fig. 3 shows the complete representation used for cones describing parts of free space. There is a straight spine, parameterized over the range $t \in [0, l]$ where l is the length of the cone. If the sides of the cone are not parallel to the spin, then $t = 0$ corresponds to the wider end.

On both the left and right the maximal radii achieved over the length of the cone occurs at $t = 0$, and are denoted b_l and b_r, respectively, describing the "big" end of the cone. The minimal radii achieved occur at $t = l$ and are denoted s_l and s_r, describing the "small" end of the cone. If $b_l = s_l$ and $b_r = s_r$, then the two sides of the cone are parallel to the spine. If not, then there is a symmetric thinning of the cone (which may start, and end, at different values of t on the left and right), where the left and right radii of the thinning parts of the cone are both given by the expression $mt + c$ where m and c are constants. Note that it is always the case that $m \leqslant 0$ and $c \geqslant 0$.

In summary the seven constants l, b_l, b_r, s_l, s_r, m, and c completely specify the shape and size of the generalized cone. In addition its location and orientation must be determined concurrently with computing these parameters.

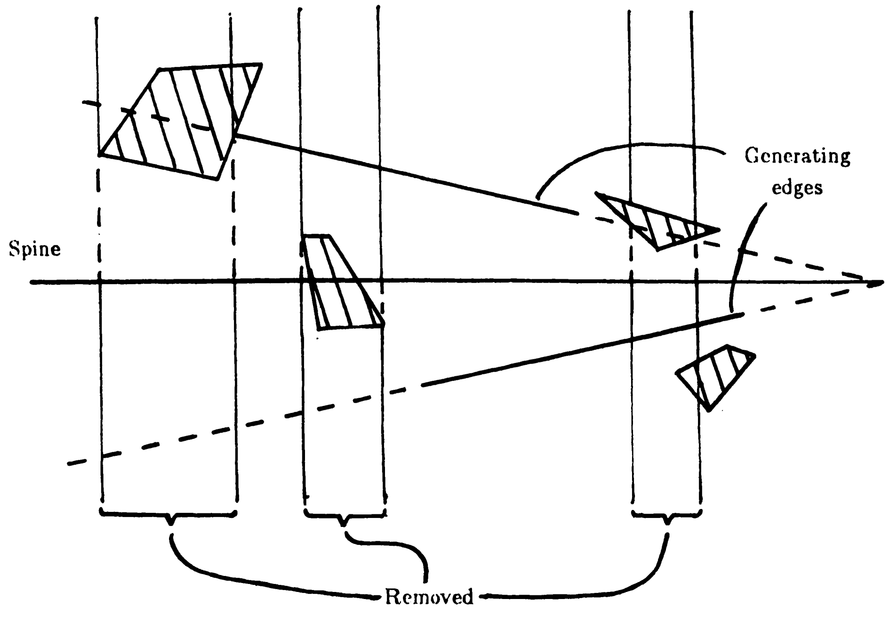

Fig. 2. Slices of generalized cone removed due to obstacles.

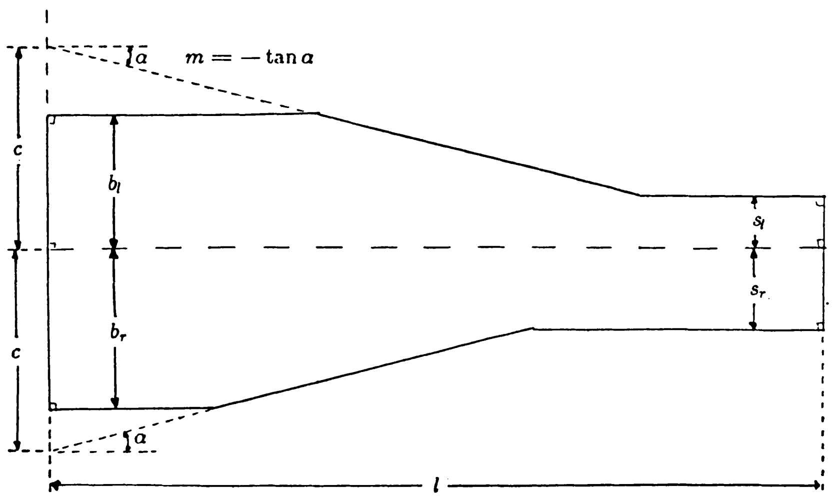

Fig. 3. Generalized cone defined by l, b_l, b_r, s_l, s_r, m, and c.

III. Determining Legal Orientations

Let the moving object be called A. It is a convex polygon and has vertices $a_1, a_2, \cdots, a_n$. Choose an origin and x and y axes. Let d_i be the distance of vertex a_i from the origin. For optimal performance of the find-path algorithm, the origin should be chosen so that $\max_{1 \leq i \leq n} d_i$ is minimized over possible origins. The direction of the x axis can be chosen arbitrarily.

Orientations relative to A are defined in terms of an angle relative to A's x axis. Thus a point's angle relative to A is the angle made by a ray from the origin to the point, relative to the x axis. A direction relative to A is the angle made by a vector in that direction with A's x axis. Let $\eta_1, \eta_2, \cdots, \eta_n$ be the angles of the vertices $a_1, a_2, \cdots, a_n$.

A. Radius Function of a Polygon

Consider first the problem of determining the volume swept by a polygon as it is moved in a straight line with fixed orientation. Let the object be moving in direction θ. The swept volume depends on the object's cross section perpendicular to that angle. The cross section can be broken into two components, that on the left of the direction of movement and that on the right. (See Fig. 4.)

Define a *radius function* $R(\xi)$ as the infimum of the distance from the origin that a line normal to a ray at angle ξ can be without intersecting the interior of A. (See Fig. 5). (The radius function is closely related to the support of a convex polygon, e.g., see [1].) The magnitudes of the left and right cross sections of Fig. 4 can then be denoted $R(\theta + \pi/2)$ and $R(\theta - \pi/2)$, respectively.

Fig. 6 shows the geometric construction of $R(\xi)$ for a given object A and angle ξ. The darkened outline in Fig. 7 shows the radius function R in polar coordinates for the same object. Thus R can be defined by

$$R(\xi) = \max_{1 \leq i \leq n} d_i \cos(\xi - \eta_i).$$

The major interest in functions of this form will be in their

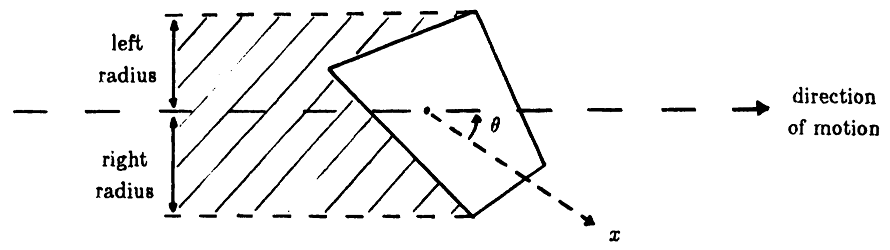

Fig. 4. Volume swept by object during pure translation.

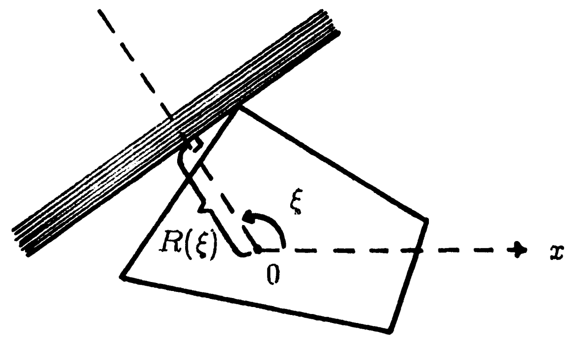

Fig. 5. Definition of $R(\xi)$.

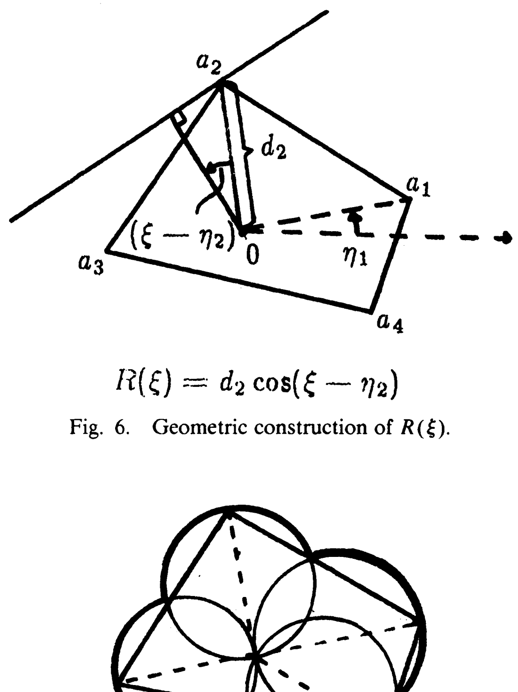

Fig. 6. Geometric construction of $R(\xi)$.

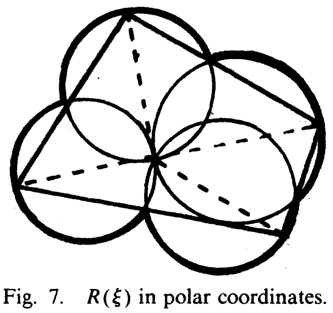

Fig. 7. $R(\xi)$ in polar coordinates.

inverse images of intervals of the form $(-\infty, r]$. Let

$$R^{\perp}(r) = \{\xi | R(\xi) \leq r\}.$$

Thus $R^{\perp}(r)$ is the range of angles at which the cross section of object A is no more than r. The inverse image can be easily computed by using the two values possible for $\arccos(r/d_i)$ for each i to form an interval containing η_i, and subtracting it from the interval $[0, 2\pi]$. We restrict our attention to moving objects which are convex polygons precisely because their radius functions can be so easily inverted.

If we write $R_{\alpha}(\xi) = R(\xi + \alpha)$, then the legal orientations of A relative to the direction of its movement down the center of a strip of diameter d can be written as

$$L = R^{\perp}_{\pi/2}(d/2) \cap R^{\perp}_{-\pi/2}(d/2).$$

Note that while L may consist of more than one interval, the set of orientations taken by A in some trajectory along the length of the cone must form a connected set—i.e., it must be a single interval, and so must be contained in a single interval of L.

Finally note that the sum of two radius functions has the same form as a radius function, and so the " $\perp$ " operation can be computed just as easily on the sums of such functions as on single functions, i.e., with R as above and

$$S(\xi) = \max_{1 \leq i \leq m} e_i \cos(\xi - \mu_i),$$

then

$$\begin{aligned}[R + S](\xi) &= R(\xi) + S(\xi) \\ &= \max_{1 \leq i \leq n, 1 \leq j \leq m} \left(d_i \cos(\xi - \eta_i) + e_j \cos(\xi - \mu_j)\right)\end{aligned}$$

and each term inside the "max" can be written in the form $f_{ij} \cos(\xi - \nu_{ij})$.

B. Bounding Polygons with an Appropriate Rectangle

In general we wish to find the legal orientations for object A when its origin is at a point with spine parameter t on the spine of a generalized cone C. A conservative set of orientations can be found by enclosing A in a rectangle which has two of its sides parallel to the spine of C. When A is oriented so that the spine has angle θ relative to it, the bounding rectangle has the following properties. Its extent forward (in the direction of increasing spine parameter) from the origin is $R(\theta)$. Its extents left and right are $R(\theta + \pi/2)$ and $R(\theta - \pi/2)$, respectively. Its extent to the rear is (rather conservatively) $d = \max_{1 \leq i \leq n} d_i$. (See Fig. 8.) The problem now is to invert this bounding operation, i.e., to find for what range of θ the bounding rectangle is within the generalized cone. That is, for each $t \in [0, l]$ we wish to find a range of valid θ, denoted $V(t)$ for orientations of the object A.

The rear of the bounding rectangle simply implies $V(t) = \phi$ outside of the interval $[d, l]$. The forward bound

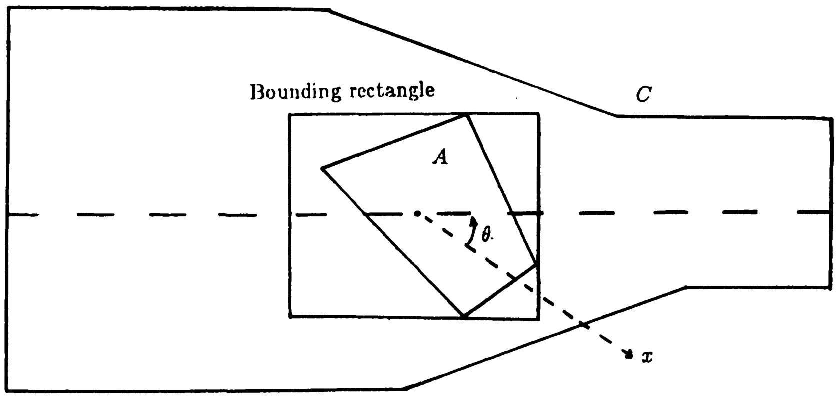

Fig. 8. Object A bounded by rectangle aligned with spine of cone C.

implies

$$V(t) \subseteq R^{\perp}(l-t).$$

The parallel sides of the "big" part of the generalized cone imply upper bounds on the width of the bounding rectangle. Thus

$$V(t) \subseteq \left(R^{\perp}(b_l) - \pi/2\right) \cap \left(R^{\perp}(b_r) + \pi/2\right).$$

Note that the terms on the right involve adding a constant to a subset of $[0, 2\pi]$ in the obvious way. Let V' be defined as

$$V'(t) = R^{\perp}(l-t) \cap \left(R^{\perp}(b_l) - \pi/2\right) \cap \left(R^{\perp}(b_r) + \pi/2\right)$$

over the interval $[d, l]$.

Subject to the above three constraints the rectangle can be as big as one which will fit in the "small" part of the generalized cone. Thus

$$\left(\left(R^{\perp}(s_l) - \pi/2\right) \cap \left(R^{\perp}(s_r) + \pi/2\right)\right) \cap V'(t) \subseteq V(t).$$

Furthermore, subject to $V'(t)$, any orientation of A is valid which results in a bounding rectangle that is within the bounds given by the decreasing boundaries of the generalized cones. Recall that at each point of the spine, with parameter t, these boundaries have both left and right radius of $mt + c$ and that $m \leqslant 0$. Thus sufficient conditions are that

$$R(\theta + \pi/2) \leqslant m \times (t + R(\theta)) + c$$
$$R(\theta - \pi/2) \leqslant m \times (t + R(\theta)) + c$$

are both true. These can be expressed as

$$\left[R_{\pi/2} - mR\right](\theta) \leqslant (mt + c)$$
$$\left[R_{-\pi/2} - mR\right](\theta) \leqslant (mt + c)$$

whence

$$\left(\left[R_{\pi/2} - mR\right]^{\perp}(mt + c) \cap \left[R_{-\pi/2} - mR\right]^{\perp}(mt + c)\right) \cap V'(t) \subseteq V(t).$$

In summary then, over the range $t \in [d, l]$, a valid subset of orientations for A can be expressed as

$$V(t) = R^{\perp}(l-t) \cap \left(R^{\perp}(b_l) - \pi/2\right) \cap \left(R^{\perp}(b_r) + \pi/2\right) \cap \Big(\left(\left(R^{\perp}(s_l) - \pi/2\right) \cap \left(R^{\perp}(s_r) + \pi/2\right)\right) \cup \Big(\left[R_{\pi/2} - mR\right]^{\perp}(mt + c) \cap \left[R_{-\pi/2} - mR\right]^{\perp}(mt + c)\Big)\Big)$$

and elsewhere $V(t) = \phi$. The key property of $V(t)$ is the following.

Lemma: For $t_1, t_2 \in [d, l]$, $t_1 \leqslant t_2$ implies $V(t_2) \subseteq V(t_1)$.

Note that $V(t)$ does not necessarily include all possible orientations for object A to be contained in cone C at point with parameter t along the spine. However it is the set of possibilities which the find-path algorithm described here is willing to consider. Its key property is that as stated by the lemma the set of valid orientations (amongst those to be considered) does not increase as the object is moved from the "big" end to the "small" end of a generalized cone.

IV. Searching for a Path

The algorithm to find a collision-free path proceeds as follows. First the generalized cones which describe free space are computed (this can be done in time $O(n^3)$). Next the cones are pairwise examined to determine if and where their spines intersect. Up to this stage the algorithm is independent of the object A to be moved through the environment. Each spine intersection point for each cone is now annotated with the subset of $[0, 2\pi]$, which describes the orientations for the object A in which it is guaranteed to be completely contained in the generalized cone. The final stage of the algorithm is to find a path from the initial position to the goal position following the spines of the generalized cones, and changing from cone to cone at spine intersection points. If the object is kept at valid orientations at each point on each spine, then the path will be collision free.

A. A Searchable Graph

A graph is built, then searched with the A^* algorithm (see Nilsson [7]). The cost function used is the distance traveled through space. The nodes of the graph consist of points on generalized cone spines which correspond to points of intersection, along with a single orientation subinterval of $[0, 2\pi]$ (except that intervals may wrap around from 2π to 0). Thus for an intersection point with parameter t on the spine of cone C, a node for each interval in $V(t)$ is built.

The arcs in the graph arise in two ways—those that correspond to transfer from one intersection point on a spine to another, and those that correspond to transfer from one spine to another at their common intersection point. All such candidate pairs of nodes are checked for connectivity. The lemma of Section III guarantees that it suffices to check if the orientation intervals of the two nodes, for intracone arcs, have a nonempty intersection. If so, then it is certainly possible to move the object A from one point to the other using any orientation within that intersection. For intercone pairs of nodes it also suffices to check for nonempty intersection of their orientation intervals, as the points are coincident in space. The graph can now be searched to find a path consisting of an ordered set of nodes from initial position to goal position.

B. Intermixing Rotations and Translations

If the graph search is successful it indicates a whole class of collision-free paths. It remains to chose one particular trajectory. The class of paths found share a common trajectory for the origin of the moving object. The orientation interval associated with each node of the path through the graph constrains the valid orientations at that point. It is also necessary to ensure that no invalid orientation is assumed during translation and rotation along the spine of a generalized cone.

It may well be the case that rotations are expensive, and it is worth searching the class of paths resulting from the graph search to find the optimal path in terms of least rotation necessary. This could be carried out using the A^* algorithm. Every node in the path found in the first search would be turned into a collection of nodes. Each pair of adjacent nodes in the path found above would have their orientation ranges intersected. Combine each end value of each intersection, and the initial and final orientations into a set of possible orientations. Each orientation which is valid for a particular path node, along with that node, generates a node in the new graph. Adjacency would be inherited from adjacency of generating nodes in the original path. The A^* algorithm is used to search this new graph.

Fig. 9 and 10 illustrate two paths found by a much simpler method. Rotations are restricted to points corresponding to nodes of the path, i.e., the object is held with fixed orientation during each translation. The orientation at each point is the midpoint of the orientations given by intersecting the allowed orientation intervals of the corresponding path node, and the subsequent node. Thus traversal of intracone arcs of the graph built above are rotation free, and traversal of intercone arcs are pure rotations. The lemma of Section III guarantees that this strategy leads to valid collision-free paths.

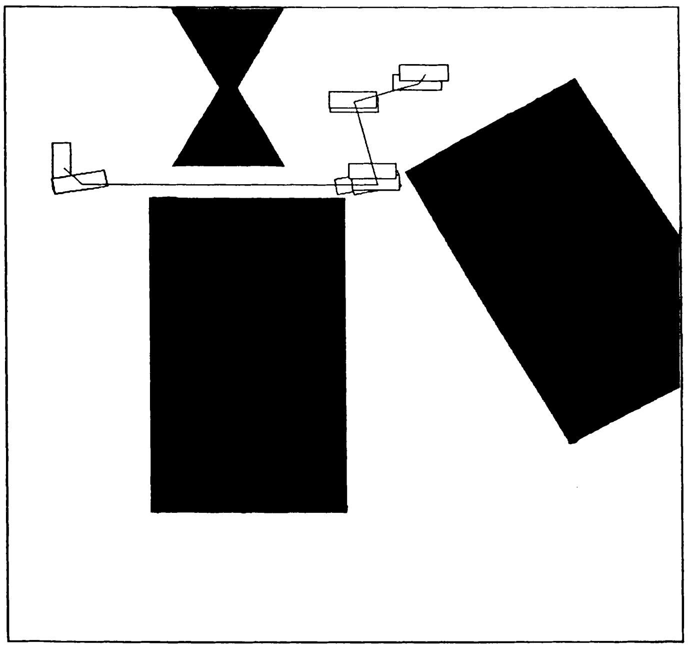
Fig. 9. Path found by algorithm.

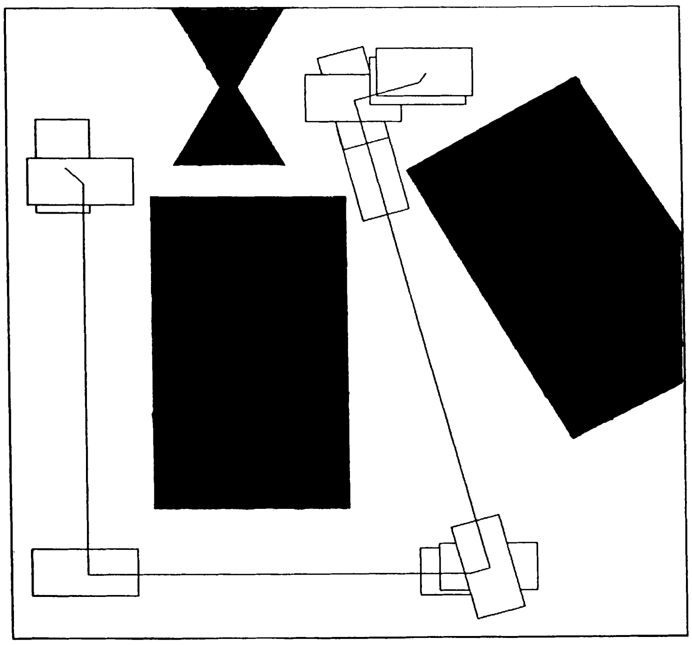
Fig. 10. Path found by algorithm.

V. Usefulness and Applicability

The major drawback of the find-path algorithm presented in this paper is that paths are restricted to follow the spines of the generalized cones chosen to represent free space. It typically does not work well in tightly constrained spaces as there are insufficient generalized cones to provide a rich choice of paths. Furthermore in a tight space the interior points of spines are always near to the spine ends. Therefore the need to extend the bounding rectangles rearward to the maximum possible distance for the moving object (so that the lemma of Section III will hold) means that typically there are very few points of intersections of generalized cones where the object's bounding rectangle in each orientation is contained within the appropriate generalized cone. This situation could be improved significantly by developing a better algorithm for decomposing free space into generalized cones. The current pruning technique is often too drastic; if an obstacle intersects a generalized cone it may be better to simply reduce the cone radius, rather than the current practice of slicing out all of the spine onto which the obstacle projects.

A. Advantages

In relatively uncluttered environments the algorithm is extremely fast. Furthermore it has the following properties which are absent from some or all of the algorithms mentioned in Section I.

1) The paths found, and the amount of computation needed, are completely independent of the chosen world coordinate system.
2) Obstacles affect the representation of free space only in their own locality. Thus additional obstacles spatially separated from a free path found when they are not present cannot affect the ability to find that free path when they are present. (Resource limited, and approximating algorithms which cut free space into rectangloid cells often suffer from this problem.)
3) Free space is represented explicitly rather than as the complement of forbidden space.
4) The paths tend to be equally far from all objects, out in truly free space, rather than scraping as close as possible to obstacles, making the instantiation of the paths by mechanical devices subject to failure due to mechanical imperfections.

We plan to use this algorithm in conjunction with an implementation of an algorithm [3] which finds paths with rotations based on a configuration-space approach. The generalized cone method will be used to find the easy parts of the path (the other method is computationally expensive both for easy and hard paths) leaving the more expensive method for the hard parts. The algorithm presented here can also provide some direction for the hard parts as it can quickly compute all topologically possible paths.

B. Three Dimensions

Clearly the eventual goal of this work is to develop algorithms which solve the find-path problem in three dimensions, with three rotational degrees of freedom for each link of general articulated devices. In this section we discuss approaches to extending the presented algorithm to the case of a single convex polyhedron moving through a space of polyhedral obstacles.

Three-dimensional generalized cones can be constructed by considering all triples of faces of obstacle polyhedra. The construction algorithm (including intersection with obstacle polyhedra) has complexity $O(n^4)$ in the number of obstacle faces. The generalized cones so constructed will have triangular cross sections. It is not clear how to pairwise intersect the generalized cones, as in general their spines will not intersect. The "path-class" idea presented below may solve this problem.

Rather than a bounding rectangle, a right triangular prism should be used where the triangle is similar to the cross section of the generalized cone being traversed. If more than one rotational degree of freedom is assumed there may be problems inverting this bounding volume. Rather than $V(t)$ having a set of intervals as a value, it will have a set of three-dimensional polyhedra.

For both the two-dimensional and three-dimensional problems it is worth investigating using paths other than the spine through a generalized cone. Then V is parameterized in terms of t and the two end points of a whole class of paths through individual cones. This increases the complexity of the search but will lead to more paths than simply using the spine.

C. Complexity Issues

In discussions with Tomás Lozano-Pérez it became evident that there exists an $O(n^2c^{\sqrt{\log n}})$ algorithm which can find the same generalized cones as does the $O(n^3)$ algorithm presented here. The spines used for generalized cones correspond to the Voronoi boundaries between obstacles (see Drysdale [4] for a complete solution to the problem), and they can be found in $O(nc^{\sqrt{\log n}})$ time. There remains a factor of $O(n)$ to do the intersection of the hypothesized generalized cones with other obstacles. It is unlikely that such an algorithm would perform better than the $O(n^3)$ algorithm presented for practical sized problems.

The complexity of pairwise intersecting the generalized cones was not addressed in the body of the paper. An upper bound of $O(n^4)$ can easily be obtained as there are at most $O(n^2)$ generalized cones to be considered. However it seems hard to find a constant where cn^2 generalized cones can be achieved constructively as the number of obstacle polygons is increased. For any c it seems that space eventually tends to become too cluttered for all the cones to be constructed. Thus it seems likely that a better bound than $O(n^4)$ exists. (In fact the Voronoi complexity above suggests a bound of $O(n^2c^{2\sqrt{\log n}})$.)

Acknowledgment

The presentation of this paper has benefitted greatly from careful readings of drafts by J. Michael Brady and Tomás Lozano-Peréz.

References

[1] Russell V. Benson, *Euclidean Geometry and Convexity*. New York: McGraw-Hill, 1966.

[2] Thomas O. Binford, "Visual perception by computer," presented at IEEE Systems, Science, and Cybernetics Conf., Miami, FL, Dec. 1971.

[3] Rodney A. Brooks and Tomás Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," M.I.T. Tech Rep. AIM-684, in preparation, 1982.

[4] R. L. (Scot) Drysdale, "Generalized Voronoi diagrams and geometric searching," Stanford, CS Rep. STAN-CS-79-705, Stanford, CA, Jan. 1979.

[5] Tomás Lozano-Pérez, "Automatic planning of manipulator transfer movements, *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 681–698, 1981.

[6] Hans P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Stanford, Univ. Tech Rep., AIM-340, Sep. 1980.

[7] Nils J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*. New York: McGraw-Hill, 1971.

[8] Shriram M. Udupa, "Collision detection and avoidance in computer controlled manipulators," Proc. IJCAI-5, MIT, Cambridge, MA, Aug. 1977, pp. 737–748.

A Graph-Theoretic Algorithm for Hierarchical Decomposition of Dynamic Systems with Applications to Estimation and Control

VENUGOPAL PICHAI, MEMBER, IEEE, MESUT E. SEZER, MEMBER, IEEE, DRAGOSLAV D. ŠILJAK, FELLOW, IEEE

Abstract—**A graph-theoretic scheme is proposed for partitioning of dynamic systems into hierarchically ordered subsystems having independent inputs and outputs. The resulting subsystems are input–output reachable as well as structurally controllable and observable, so that a "piece-by-piece" design of estimators and controllers can be accomplished for systems with large dimensions without excessive computer requirements.**

I. Introduction

DECOMPOSITION principle stands for a variety of more or less disparate problems and techniques which are held together by a common theme: a large system of equations is broken up into subsystems which are then solved separately and put together to produce a global solution (see, for example, [1]–[3]). The actual tearing of the system may be performed in any number of ways in order to achieve computational and conceptual simplifications of the overall problem. The most efficient tearing techniques so far have been those formulated in the graph-theoretic setting because of the convenience in computer implementations. The partitioning algorithms take advantage of the inherent structure of the system to decompose it into a number of irreducible hierarchically ordered subsystems, which can subsequently be solved from top to bottom independently of each other. This approach has been used to solve a system of algebraic equations with several thousand nodes without requiring excessive computation time [4].

In order to reproduce in the dynamic systems context some of the benefits of structural insights and simplifications achieved in solving large algebraic systems, directed graphs (digraphs) have been associated with large-scale dynamic systems [5], and various stability and decentralized control problems have been solved in this way [3]. This prompted a number of efforts to introduce a hierarchical decomposition of dynamic systems using graph-theoretic methods [6]–[9], and come up with simplified stability problems. The underlying idea has been to mimic the approach used in the algebraic setting and identify the irreducible dynamic subsystems with *strong components* of the corresponding digraph. To test the sta-

Manuscript received April 13, 1982; revised September 1, 1982. This work was supported in part by the National Science Foundation under Grant ECS-8011210.

V. Pichai is with Quadrex Corporation, Campbell, CA 95008.

M. E. Sezer is with the Systems Engineering Department, University of Petroleum and Minerals, Dhahran, Saudia Arabia, on leave from the Middle East Technical University, Ankara, Turkey.

D. D. Šiljak is with the School of Engineering, University of Santa Clara, Santa Clara, CA 95053.

References

[1] Russell V. Benson, *Euclidean Geometry and Convexity.* New York: McGraw-Hill, 1966.

[2] Thomas O. Binford, "Visual perception by computer," presented at IEEE Systems, Science, and Cybernetics Conf., Miami, FL, Dec. 1971.

[3] Rodney A. Brooks and Tomás Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," M.I.T. Tech Rep. AIM-684, in preparation, 1982.

[4] R. L. (Scot) Drysdale, "Generalized Voronoi diagrams and geometric searching," Stanford, CS Rep. STAN-CS-79-705, Stanford, CA, Jan. 1979.

[5] Tomás Lozano-Pérez, "Automatic planning of manipulator transfer movements, *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 681–698, 1981.

[6] Hans P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Stanford, Univ. Tech Rep., AIM-340, Sep. 1980.

[7] Nils J. Nilsson, *Problem-Solving Methods in Artificial Intelligence.* New York: McGraw-Hill, 1971.

[8] Shriram M. Udupa, "Collision detection and avoidance in computer controlled manipulators," Proc. IJCAI-5, MIT, Cambridge, MA, Aug. 1977, pp. 737–748.

A Graph-Theoretic Algorithm for Hierarchical Decomposition of Dynamic Systems with Applications to Estimation and Control

VENUGOPAL PICHAI, MEMBER, IEEE, MESUT E. SEZER, MEMBER, IEEE, DRAGOSLAV D. ŠILJAK, FELLOW, IEEE

Abstract—**A graph-theoretic scheme is proposed for partitioning of dynamic systems into hierarchically ordered subsystems having independent inputs and outputs. The resulting subsystems are input–output reachable as well as structurally controllable and observable, so that a "piece-by-piece" design of estimators and controllers can be accomplished for systems with large dimensions without excessive computer requirements.**

I. Introduction

DECOMPOSITION principle stands for a variety of more or less disparate problems and techniques which are held together by a common theme: a large system of equations is broken up into subsystems which are then solved separately and put together to produce a global solution (see, for example, [1]–[3]). The actual tearing of the system may be performed in any number of ways in order to achieve computational and conceptual simplifications of the overall problem. The most efficient tearing techniques so far have been those formulated in the graph-theoretic setting because of the convenience in computer implementations. The partitioning algorithms take advantage of the inherent structure of the system to decompose it into a number of irreducible hierarchically ordered subsystems, which can subsequently be solved from top to bottom independently of each other. This approach has been used to solve a system of algebraic equations with several thousand nodes without requiring excessive computation time [4].

In order to reproduce in the dynamic systems context some of the benefits of structural insights and simplifications achieved in solving large algebraic systems, directed graphs (digraphs) have been associated with large-scale dynamic systems [5], and various stability and decentralized control problems have been solved in this way [3]. This prompted a number of efforts to introduce a hierarchical decomposition of dynamic systems using graph-theoretic methods [6]–[9], and come up with simplified stability problems. The underlying idea has been to mimic the approach used in the algebraic setting and identify the irreducible dynamic subsystems with *strong components* of the corresponding digraph. To test the sta-

Manuscript received April 13, 1982; revised September 1, 1982. This work was supported in part by the National Science Foundation under Grant ECS-8011210.

V. Pichai is with Quadrex Corporation, Campbell, CA 95008.

M. E. Sezer is with the Systems Engineering Department, University of Petroleum and Minerals, Dhahran, Saudia Arabia, on leave from the Middle East Technical University, Ankara, Turkey.

D. D. Šiljak is with the School of Engineering, University of Santa Clara, Santa Clara, CA 95053.