'90 KACC 1990. 10. 26~27

Motion Planning with Planar Geometric Models*

Myung-Soo Kim, Sang-Ryong Moon, and *Kwan-Hee Lee*

Department of Computer Science
POSTECH
P.O. Box 125, Pohang 790-600, Korea.
TEL:(0562)-79-2249 FAX:(0562)-79-2299

Abstract

We present algebraic algorithms for collision-avoidance robot motion planning problems with planar geometric models. By decomposing the collision-free space into *horizontal vertex visibility cells* and connecting these cells into a connectivity graph, we represent the global topological structure of collision-free space. Using the *C-space* obstacle boundaries and this connectivity graph we generate exact (non-heuristic) *compliant* and *gross* motion paths of planar curved objects moving with a fixed orientation amidst similar obstacles. The *gross* motion planning algorithm is further extended (though using approximations) to the case of objects moving with both translational and rotational degrees of freedom by taking slices of the overall orientations into finite segments.

1 Introduction

We consider collision-avoidance robot motion planning problems for planar robots bounded by algebraic curve segments moving amidst planar stationary obstacles similarly defined. Though there have been various algorithms developed for planning robot motions avoiding collisions with obstacles, most of the results are limited to the case of polygonal and/or polyhedral moving objects and obstacles. The general robot motion planning algorithms developed by Schwartz and Sharir [21] and recently improved by Canny [8] essentially consider point motions in the *C-space* (*Configuration space*) where the *C-space* obstacles are represented by semi-algebraic sets. For polygonal and/or polyhedral robot motions the algebraic constraints defining the *C-space* obstacle boundaries have no singularities on their defining hyper-surfaces. Each connected component of the *C-space* obstacles can be represented as a conjunction of the corresponding algebraic inequalities, i.e., as a semi-algebraic set. However, for the more general complicated robots bounded by algebraic curve segments and surface patches, it is not easy to represent the *C-space* obstacles in terms of semi-algebraic sets even for the simple translatory motions with fixed orientations. This is because the corresponding *C-space* obstacle boundaries usually have very high degree curves and surfaces. Thus, there may occur very complicated singularities on the boundary. Representing these complicated algebraic varieties with singularities as semi-algebraic sets is still a difficult open problem due to the current status of geometric modeling research. Thus, we represent the *C-space* obstacles in boundary representations and develop appropriate motion planning algorithms for them. We assume the previous algorithms to generate the *C-space* obstacles in boundary representations for both moving objects and obstacles bounded by algebraic curves and algebraic surfaces, [2,3,4], and the object motions restricted to the translatory motions with fixed orientations.

*Research supported in part by RIST grant 90-0244F.

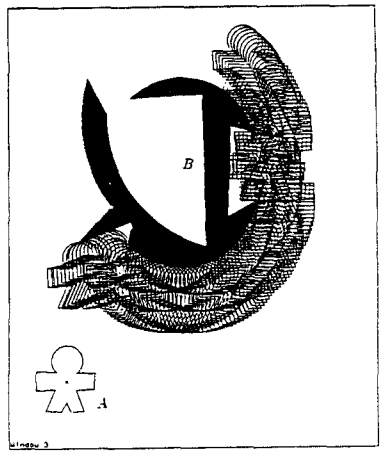


Figure 1: Compliant Motion.

For the motions of planar robots with fixed orientations, we develop *exact* algorithms for both *compliant* and *gross* motion plannings. Being *exact* means the algorithms we develop are non-heuristic. The robots are in *compliant* motions when they move with continuous contacts with the obstacles, see Figure 1. This is equivalent to moving the corresponding reference points along the *C-space* obstacle boundaries. Thus, each compliant motion path can be obtained by doing a search from the start point to the goal point on the graph representing the *C-space* obstacle boundaries. Note that each connected component of the *C-space* obstacles has its boundary representation as a circular list of algebraic curve segments. For collision-avoidance *gross*

motion plannings, we first represent the collision-free space efficiently by decomposing the free space into the horizontal vertex visibility cells (*hv-cells*) and connecting these cells into a connectivity graph (*hv-cells-graph*). We can compute this decomposition by doing a *plane sweep* over the *C-space* obstacles, see [5,6]. This graph has $O(n)$ space complexity and represents a global topological structure for the collision-free space. We then locate the start and goal nodes of the hv-cells-graph corresponding to the hv-cells containing the start and goal positions of the reference point respectively. We do a search on this graph and construct a connected path from the start node to the goal node. This path gives a connected sequence of hv-cells along which the robot can move without collisions with obstacles. The path also represents a topological structure for the class of collision-free paths of the robot passing through the channel determined by this connected sequence of hv-cells. We construct a corresponding sequence of algebraic curve segments imbedded in this channel and also connecting the start and the goal points. In the construction of these curve segments we can take further considerations on the local configurations of the connected hv-cells. Various heuristic techniques may be applied.

In contrast to the traditional free-space representations using triangulations of the polygonal *C-spaces*, we use the hv-cells decompositions which can easily be constructed by doing a plane sweep over the *C-space* obstacles. Triangulating planar curved geometric models into similar triangular shapes bounded by three curve segments is not easy in general. Further, the edge orientations of these triangular cells may not be uniform. These difficulties and shape irregularities may prohibit efficient implementations. However, in hv-cells, the edges on the left and right sides are curve segments from the original *C-space* obstacle boundaries, and the extra edges added are always horizontal edges. Thus, it is easy to classify all the possible configurations of the connections between adjacent hv-cells. To minimize the number of cells we can take the decompositions only at the y-extreme vertices, not at every vertex. In this case, the left and right sides of each hv-cell can be sequences of connected curve segments in general. Though with these simplicities and efficiencies of hv-cells, there are also various disadvantages, too. Due to its strong dependancy on the horizontal direction, many thin horizontal cells may result. An extreme case is when two y-extreme vertices have the same y-coordinates and the corresponding hv-cell degenerates into a line segment. However, in these special cases, it should not be very difficult to take further considerations on the vertical directions and get around these difficulties by using some other compatible techniques.

The *gross* motion planning algorithm is extended (though using approximations) to the object motions with both translations and rotations by using the traditional slicing method, [16,17], which takes finite slices of the overall orientations. For this extension we use our previous algorithm to compute the *rotational sweep volumes* of planar geometric models, [12]. Note that in the slicing method of [16] the rotational sweep volumes of polygonal moving objects, having circular arcs and line segments on their boundaries, are further approximated by polygons totally enclosing the sweep volumes. With our generalizations to the curved case, we can directly use the curved rotational sweep volumes in the *C-space* obstacle generations. For the polygons moving with translations and rotations amidst polygonal obstacles, Avnaim, Boissonnat, and Faverjon [1] presents an exact algorithm to generate the corresponding *C-space* obstacles with ruled surface patches on their boundaries. To generate gross motion paths, they represent the collision-free space as a collection of triangular prism cells. These prism cells are constructed using a space-sweep method which sweeps a plane along the θ-axis while maintaining the triangulation of polygonal free-space. The space sweep updates the triangulation whenever a triangle degenerates into a line segment or any two polygonal regions merge into a new polygonal region. Since this algorithm uses various features of polygons, it is not easy to extend this result to the curved case. The simple structure of hv-cells would make this generalization easier than using triangular prisms. Degeneracy detection for hv-cells would be easier than triangles since the degeneracy for hv-cell occurs when two horizontal lines come to overlap whereas the triangular degeneracy occurs when any two of the three edges overlap. The high time complexity $O(m^6n^6\alpha(mn))$ of Avnaim, Boissonnat, and Faverjon [1] seems due to the difficulties in detecting the degeneracies of the triangles maintained in the space-sweep. One possible way to reduce this complexity would be to simplify the degeneracy detections by using simpler cells like hv-cells rather than the triangular cells for free-space decompositions.

The rest of the paper is as follows. In §2 we review the previous computational geometry algorithms for planar geometric models useful in designing efficient motion planning algorithms in this paper. These include the algorithms to compute the convex hulls and various decompositions of planar geometric models. In §3 we present various collision-avoidance motion planning algorithms for planar geometric models. Finally, in §4 we conclude this paper. We implemented these motion planning algorithms on Symbolics 3650 Lisp Machine and SUN4/330GX Sparc Station using Common Lisp for the simple planar geometric models bounded by circular arcs and line segments. Further implementations are currently going on using Gaussian Approximations [13] to deal with planar geometric models bounded by general planar algebraic curve segments.

2 Algorithms for Planar Geometric Models

We review some of the previous algorithms designed for planar geometric models. These algorithms are useful in designing efficient algorithms in later sections. In §2.1–2.2 we review the algorithms to compute the convex hulls and various decompositions of planar geometric models. For gross motion planning, one may compute the *C-Space* obstacles for the convex hulls of the moving objects and the obstacles. Alternatively, one may first decompose the given models into certain primitive pieces and then apply the *C-space* obstacle generations on the decomposed pieces of object and obstacle pairs. In §2.3 we also consider the Voronoi Diagrams of planar geometric models and derive systems of polynomial equations defining the Voronoi edges. Under the Euclidean distance function the degrees of Voronoi edges are extremely high. Thus, it is not plausible to apply the retraction method of [18] directly to the *C-space* obstacles. Instead, in §3.1 we present how to use the Voronoi Diagrams of the outer and simple carrier polygons of *C-space* obstacles to compute collision-avoidance paths.

2.1 Convex Hulls

The convex hull computation is a fundamental one in computational geometry. There are numerous applications in which the convex hulls of complex objects can be used effectively to make certain geometric decisions easier. For example, a null intersection between the convex hulls of two objects implies a null intersection between the original objects. Since intersection testing for convex objects is easier than for general objects, convex hulls intersection is often used as an efficient first test in an intersection detection algorithm for non-convex objects. Addi-

tional motivation arises from the use of convex hulls for heuristic collision-free motion planning of general objects among obstacles. Motion planning is easier for convex objects and obstacles, [3].

Several linear–time algorithms for computing the convex hulls of simple planar polygons are known, [11,14]. These algorithms achieve the more efficient $O(n)$ bound whereas the $\Omega(n \log n)$ lower bound applies to the general problem of computing the convex hull of n points in the plane, [19]. By generalizing [11] to an edge–based algorithm Schäffer and Van Wyk [20] extend the planar polygon results to a linear–time algorithm for curved objects bounded by piecewise–smooth Jordan curves. Dobkin and Souvaine [10] suggests a linear–time convex hull algorithm for a class of planar curved objects. Bajaj and Kim [5,6] also proposed a simple linear time algorithm for computing the convex hulls of objects bounded by algebraic curves.

2.2 Decompositions

Most algorithms in computational geometry have been designed for discrete objects like points, lines, polygons, and polyhedra, see [15,19]. Using the *splinegon* as a generalization of the polygon, Dobkin and Souvaine [10] presents methods for extending polygonal algorithms to algorithms for splinegons. A *splinegon* is a polygon whose edges have been replaced by convex, concave or linear curve segments, and the *carrier* polygon of a splinegon is the polygon connecting adjacent vertices of a splinegon. Dobkin, Souvaine, and Van Wyk [9] show that the $O(n \log\log n)$ time algorithm for the horizontal-vertex-visibility partition of a simple polygon [22] can easily be generalized to a simple splinegon, and using this partition they present an algorithm to decompose a simple splinegon into a union of monotone pieces and further into a union of differences of unions of possibly overlapping convex pieces. They also show that simplifying the carrier polygon can be quite expensive by constructing an n-sided splinegon whose smallest simple carrier polygon has $\Omega(n^2)$ edges.

Bajaj and Kim [5,6] presented an $O(n \log\log n + k \cdot d^{O(1)})$ algorithm to compute a simple *carrier* polygon of planar geometric models, where n is the number of monotone boundary curve segments and k is the number of edges in the resulting simple carrier polygon. Further, the worst case upper bound of k is shown to be the optimal $\Theta(n^2)$. Bajaj and Kim [5,6] also presented an $O(n \log\log n + K \cdot d^{O(1)})$ algorithm to construct a simple *characteristic* carrier polygon of planar curved object, where K is the minimum number of edges for (possibly non-simple) characteristic carrier polygons of the object, see Figure 2(a). A carrier polygon is *characteristic* if it differs from the original object by convex regions each of which is totally contained in the interior of the object or in its exterior.

We can also construct within the same time bound $O(n \log\log n + K \cdot d^{O(1)})$, an *inner* polygon (resp. an *outer* polygon) which is a simple polygon totally contained in (resp. totally containing) the object, see Figure 2(b)–2(c). In contrast to the simple carrier polygon construction, the worst-case upper bound for K can be arbitrarily large as the inner angle between two adjacent edges approaches to 0 or 2π, however, it is small in practice. K (henceforth called the *characteristic* number) in some sense represents the shape degeneracy of the object. In the construction of the characteristic, inner and outer polygons, we assume the object has no vertex with its inner angle being 0 or 2π. Using these polygons, we can compute (1) a convex decomposition of the object as a difference of unions of disjoint convex objects, and (2) a decomposition of the object as a union of disjoint certain primitive objects.

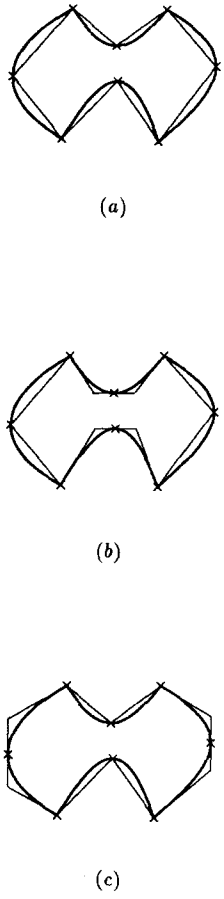


Figure 2: (a) Simple Characteristic Polygon. (b) Inner Polygon. (c) Outer Polygon.

2.3 Voronoi Diagrams of Algebraic Curves

Yap [24] gives an efficient $O(n \log n)$ algorithm to construct the Voronoi Diagram for a set of points, line segments, and circular arcs in the plane. The Voronoi edges consist of conic curves in this case. However, in the Voronoi Diagrams for general algebraic curve segments, we may have Voronoi edges of very high degree algebraic curve segments. In the following, we derive the algebraic equations of the bisectors for (1) a single algebraic curve $f = 0$, and (2) two different irreducible algebraic curves $f = 0$ and $g = 0$. By Bezout theorem [23], we can show the degrees of these bisectors can be as high as $O(d^4)$ when $f = 0$ and $g = 0$ are of degree d. Since the *C-space* obstacle boundary curves can have degree $O(d^3)$ for moving objects and obstacles bounded by degree d algebraic curves, we may have $O(d^{12})$ degree curve segments on the corresponding Voronoi Diagrams of the *C-space* obstacles. This degree complexity justifies our alternative approximation-based approaches to be discussed in §3.1 using the inner and outer polygons of the *C-space* obstacles.

motion plannings, we first represent the collision-free space efficiently by decomposing the free space into the horizontal vertex visibility cells (*hv-cells*) and connecting these cells into a connectivity graph (*hv-cells-graph*). We can compute this decomposition by doing a *plane sweep* over the *C-space* obstacles, see [5,6]. This graph has $O(n)$ space complexity and represents a global topological structure for the collision-free space. We then locate the start and goal nodes of the hv-cells-graph corresponding to the hv-cells containing the start and goal positions of the reference point respectively. We do a search on this graph and construct a connected path from the start node to the goal node. This path gives a connected sequence of hv-cells along which the robot can move without collisions with obstacles. The path also represents a topological structure for the class of collision-free paths of the robot passing through the channel determined by this connected sequence of hv-cells. We construct a corresponding sequence of algebraic curve segments imbedded in this channel and also connecting the start and the goal points. In the construction of these curve segments we can take further considerations on the local configurations of the connected hv-cells. Various heuristic techniques may be applied.

In contrast to the traditional free-space representations using triangulations of the polygonal *C-spaces*, we use the hv-cells decompositions which can easily be constructed by doing a plane sweep over the *C-space* obstacles. Triangulating planar curved geometric models into similar triangular shapes bounded by three curve segments is not easy in general. Further, the edge orientations of these triangular cells may not be uniform. These difficulties and shape irregularities may prohibit efficient implementations. However, in hv-cells, the edges on the left and right sides are curve segments from the original *C-space* obstacle boundaries, and the extra edges added are always horizontal edges. Thus, it is easy to classify all the possible configurations of the connections between adjacent hv-cells. To minimize the number of cells we can take the decompositions only at the y-extreme vertices, not at every vertex. In this case, the left and right sides of each hv-cell can be sequences of connected curve segments in general. Though with these simplicities and efficiencies of hv-cells, there are also various disadvantages, too. Due to its strong dependancy on the horizontal direction, many thin horizontal cells may result. An extreme case is when two y-extreme vertices have the same y-coordinates and the corresponding hv-cell degenerates into a line segment. However, in these special cases, it should not be very difficult to take further considerations on the vertical directions and get around these difficulties by using some other compatible techniques.

The *gross* motion planning algorithm is extended (though using approximations) to the object motions with both translations and rotations by using the traditional slicing method, [16,17], which takes finite slices of the overall orientations. For this extension we use our previous algorithm to compute the *rotational sweep volumes* of planar geometric models, [12]. Note that in the slicing method of [16] the rotational sweep volumes of polygonal moving objects, having circular arcs and line segments on their boundaries, are further approximated by polygons totally enclosing the sweep volumes. With our generalizations to the curved case, we can directly use the curved rotational sweep volumes in the *C-space* obstacle generations. For the polygons moving with translations and rotations amidst polygonal obstacles, Avnaim, Boissonnat, and Faverjon [1] presents an exact algorithm to generate the corresponding *C-space* obstacles with ruled surface patches on their boundaries. To generate gross motion paths, they represent the collision-free space as a collection of triangular prism cells. These prism cells are constructed using a space-sweep method which sweeps a plane along the θ-axis while maintaining the triangulation of polygonal free-space. The space sweep updates the triangulation whenever a triangle degenerates into a line segment or any two polygonal regions merge into a new polygonal region. Since this algorithm uses various features of polygons, it is not easy to extend this result to the curved case. The simple structure of hv-cells would make this generalization easier than using triangular prisms. Degeneracy detection for hv-cells would be easier than triangles since the degeneracy for hv-cell occurs when two horizontal lines come to overlap whereas the triangular degeneracy occurs when any two of the three edges overlap. The high time complexity $O(m^6n^6\alpha(mn))$ of Avnaim, Boissonnat, and Faverjon [1] seems due to the difficulties in detecting the degeneracies of the triangles maintained in the space-sweep. One possible way to reduce this complexity would be to simplify the degeneracy detections by using simpler cells like hv-cells rather than the triangular cells for free-space decompositions.

The rest of the paper is as follows. In §2 we review the previous computational geometry algorithms for planar geometric models useful in designing efficient motion planning algorithms in this paper. These include the algorithms to compute the convex hulls and various decompositions of planar geometric models. In §3 we present various collision-avoidance motion planning algorithms for planar geometric models. Finally, in §4 we conclude this paper. We implemented these motion planning algorithms on Symbolics 3650 Lisp Machine and SUN4/330GX Sparc Station using Common Lisp for the simple planar geometric models bounded by circular arcs and line segments. Further implementations are currently going on using Gaussian Approximations [13] to deal with planar geometric models bounded by general planar algebraic curve segments.

2 Algorithms for Planar Geometric Models

We review some of the previous algorithms designed for planar geometric models. These algorithms are useful in designing efficient algorithms in later sections. In §2.1–2.2 we review the algorithms to compute the convex hulls and various decompositions of planar geometric models. For gross motion planning, one may compute the *C-Space* obstacles for the convex hulls of the moving objects and the obstacles. Alternatively, one may first decompose the given models into certain primitive pieces and then apply the *C-space* obstacle generations on the decomposed pieces of object and obstacle pairs. In §2.3 we also consider the Voronoi Diagrams of planar geometric models and derive systems of polynomial equations defining the Voronoi edges. Under the Euclidean distance function the degrees of Voronoi edges are extremely high. Thus, it is not plausible to apply the retraction method of [18] directly to the *C-space* obstacles. Instead, in §3.1 we present how to use the Voronoi Diagrams of the outer and simple carrier polygons of *C-space* obstacles to compute collision-avoidance paths.

2.1 Convex Hulls

The convex hull computation is a fundamental one in computational geometry. There are numerous applications in which the convex hulls of complex objects can be used effectively to make certain geometric decisions easier. For example, a null intersection between the convex hulls of two objects implies a null intersection between the original objects. Since intersection testing for convex objects is easier than for general objects, convex hulls intersection is often used as an efficient first test in an intersection detection algorithm for non-convex objects. Addi-

tional motivation arises from the use of convex hulls for heuristic collision-free motion planning of general objects among obstacles. Motion planning is easier for convex objects and obstacles, [3].

Several linear–time algorithms for computing the convex hulls of simple planar polygons are known, [11,14]. These algorithms achieve the more efficient $O(n)$ bound whereas the $\Omega(n \log n)$ lower bound applies to the general problem of computing the convex hull of n points in the plane, [19]. By generalizing [11] to an edge–based algorithm Schäffer and Van Wyk [20] extend the planar polygon results to a linear–time algorithm for curved objects bounded by piecewise–smooth Jordan curves. Dobkin and Souvaine [10] suggests a linear–time convex hull algorithm for a class of planar curved objects. Bajaj and Kim [5,6] also proposed a simple linear time algorithm for computing the convex hulls of objects bounded by algebraic curves.

2.2 Decompositions

Most algorithms in computational geometry have been designed for discrete objects like points, lines, polygons, and polyhedra, see [15,19]. Using the *splinegon* as a generalization of the polygon, Dobkin and Souvaine [10] presents methods for extending polygonal algorithms to algorithms for splinegons. A *splinegon* is a polygon whose edges have been replaced by convex, concave or linear curve segments, and the *carrier* polygon of a splinegon is the polygon connecting adjacent vertices of a splinegon. Dobkin, Souvaine, and Van Wyk [9] show that the $O(n \log \log n)$ time algorithm for the horizontal-vertex-visibility partition of a simple polygon [22] can easily be generalized to a simple splinegon, and using this partition they present an algorithm to decompose a simple splinegon into a union of monotone pieces and further into a union of differences of unions of possibly overlapping convex pieces. They also show that simplifying the carrier polygon can be quite expensive by constructing an n-sided splinegon whose smallest simple carrier polygon has $\Omega(n^2)$ edges.

Bajaj and Kim [5,6] presented an $O(n \log \log n + k \cdot d^{O(1)})$ algorithm to compute a simple *carrier* polygon of planar geometric models, where n is the number of monotone boundary curve segments and k is the number of edges in the resulting simple carrier polygon. Further, the worst case upper bound of k is shown to be the optimal $\Theta(n^2)$. Bajaj and Kim [5,6] also presented an $O(n \log \log n + K \cdot d^{O(1)})$ algorithm to construct a simple *characteristic* carrier polygon of planar curved object, where K is the minimum number of edges for (possibly non-simple) characteristic carrier polygons of the object, see Figure 2(a). A carrier polygon is *characteristic* if it differs from the original object by convex regions each of which is totally contained in the interior of the object or in its exterior.

We can also construct within the same time bound $O(n \log \log n + K \cdot d^{O(1)})$, an *inner* polygon (resp. an *outer* polygon) which is a simple polygon totally contained in (resp. totally containing) the object, see Figure 2(b)–2(c). In contrast to the simple carrier polygon construction, the worst-case upper bound for K can be arbitrarily large as the inner angle between two adjacent edges approaches to 0 or 2π, however, it is small in practice. K (henceforth called the *characteristic* number) in some sense represents the shape degeneracy of the object. In the construction of the characteristic, inner and outer polygons, we assume the object has no vertex with its inner angle being 0 or 2π. Using these polygons, we can compute (1) a convex decomposition of the object as a difference of unions of disjoint convex objects, and (2) a decomposition of the object as a union of disjoint certain primitive objects.

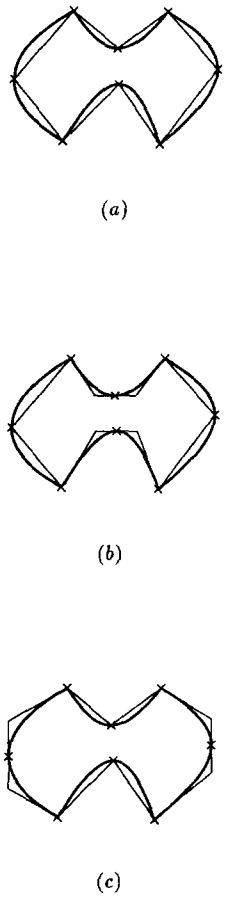


Figure 2: (a) Simple Characteristic Polygon. (b) Inner Polygon. (c) Outer Polygon.

2.3 Voronoi Diagrams of Algebraic Curves

Yap [24] gives an efficient $O(n \log n)$ algorithm to construct the Voronoi Diagram for a set of points, line segments, and circular arcs in the plane. The Voronoi edges consist of conic curves in this case. However, in the Voronoi Diagrams for general algebraic curve segments, we may have Voronoi edges of very high degree algebraic curve segments. In the following, we derive the algebraic equations of the bisectors for (1) a single algebraic curve $f = 0$, and (2) two different irreducible algebraic curves $f = 0$ and $g = 0$. By Bezout theorem [23], we can show the degrees of these bisectors can be as high as $O(d^4)$ when $f = 0$ and $g = 0$ are of degree d. Since the *C-space* obstacle boundary curves can have degree $O(d^3)$ for moving objects and obstacles bounded by degree d algebraic curves, we may have $O(d^{12})$ degree curve segments on the corresponding Voronoi Diagrams of the *C-space* obstacles. This degree complexity justifies our alternative approximation-based approaches to be discussed in §3.1 using the inner and outer polygons of the *C-space* obstacles.

2.3.1 Bisector of an Algebraic Curve

Let C be an algebraic curve defined by an implicit equation $f = 0$, then the bisector of C is the set of points which are equidistant from two different points on C. Suppose p is equidistant from p_1 and p_2 on C, i.e., $d(p,p_1) = d(p,p_2) = r$ for some $r > 0$, then the circle of radius r with center at p is tangent to C at p_1 and p_2. Further, p is a singular point of the constant radius offset curve of C with respect to a radius r. The algebraic equation $F(x,y,r) = 0$ for this offset curve can be derived from [3]. Since p is a singular point of this curve, $p = (x,y)$ satisfies $F = F_x = F_y = 0$. By eliminating the variable r from any two of these three equations, we can derive three algebraic equations $S_r(F,F_x), S_r(F,F_y), S_r(F_x,F_y)$, where S_r is the Sylvester resultant eliminating the variable r. The bisector curve of C satisfies these three S_r equations simultaneously and thus the common factor of the three S_r's. Since F has algebraic degree $O(d^2)$ and S_r's have degree $O(d^4)$, the bisector of C may have degree $O(d^4)$ in the worst case, [23].

2.3.2 Bisector of Two Algebraic Curves

Let C and D be two different algebraic curve segments defined by implicit equations $f = 0$ and $g = 0$ respectively, then the bisector of C and D is the set of points which are equidistant from two different points p_1 and p_2 on C and D respectively. The bisector equation is given by the following equation. Thus, the bisector can have degree $O(d^4)$ in the worst case, [23].

$$\begin{cases} f(x_1,y_1) = 0 \text{ and } p_1 = (x_1,y_1) \in C & (1) \\ g(x_2,y_2) = 0 \text{ and } p_2 = (x_2,y_2) \in D & (2) \\ f_x \cdot \beta_1 - f_y \cdot \alpha_1 = 0 & (3) \\ g_x \cdot \beta_2 - g_y \cdot \alpha_2 = 0 & (4) \\ \alpha_1^2 + \beta_1^2 = \alpha_2^2 + \beta_2^2 & (5) \end{cases}$$

3 Motion Planning with Planar Geometric Models

In this section, we consider various applications of the computational geometry algorithms presented in §2 to the collision-free robot motion planning problems for planar geometric models. We assume the *C-space* obstacles [3] are computed for the robots moving with fixed orientations and only consider point motions among the *C-space* obstacles.

3.1 Voronoi Diagrams for Outer and Simple Carrier Polygons

As discussed in §2.2, we can construct disjoint outer polygons of multiple disjoint planar *C-space* obstacles within $O(n \log n + K \cdot d^{O(1)})$ time, where K is the characteristic number of the *C-space* obstacles. The collision-free space of these outer polygons is continuously deformable to the free space of original obstacles. Since each outer polygonal edge and its corresponding obstacle boundary edge are horizontally visible from each other, a point p which is inside an outer polygon, but not in any obstacle, can move along a horizontal line segment into an outer polygonal edge without colliding with any obstacle. Thus, we may assume the start point p_S and the goal point p_G are in the free space of outer polygons. Now, we can use the method of [18] using the Voronoi diagram of outer polygons to construct a collision-free path γ from p_S to p_G. Since the outer polygons of obstacles have total $O(K)$ number of edges, this method is attractive when K is almost linear or reasonably small.

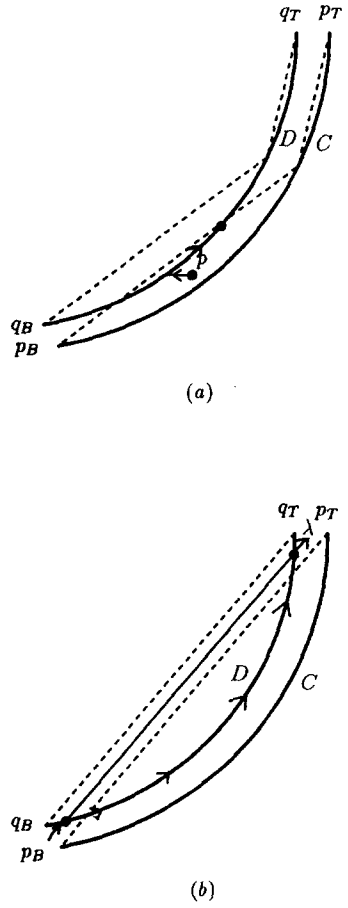


Figure 3: (a) Moving p Along D. (b) Moving p Along Colliding Convex Edge.

We can construct disjoint simple carrier polygons of disjoint obstacles within $O(n \log n + k \cdot d^{O(1)})$ time, where k is the total number of edges in the carrier polygons, which is at most $O(n^2)$. We first consider how to move a point p which is inside a carrier polygon, but not in any obstacle, without colliding with obstacles. Let H be the horizontal visibility cell containing the point p and in the exterior of obstacles. Assume C and D are the right and left sides of H. Since p is inside a carrier polygon, either C or D is concave. We may assume C is concave and p is contained in the convex region $S(C)$. We move p until it hits a carrier polygonal edge or the left side D. If it hits D first, then we can move p along D until it hits a carrier polygonal edge, see Figure 3(a). Thus, we may assume the start point p_S and the goal point p_G are in the free space of the carrier polygons. Then a Voronoi diagram of these disjoint simple carrier polygons and a path avoiding collisions with these carrier polygons can be constructed within $O(k \log k)$ time, [18]. Though this path may intersect with some obstacles, this collision could occur only with convex edges of the obstacle, see Figure 3(b).

Since we assume each convex edge is monotone, it is obvious in which direction we have to move along the colliding convex edge. Further the construction of a simple carrier polygon is more efficient than the construction of an outer polygon in the worst case. As long as the robots are allowed to move with contacts with the obstacles, this method would be more efficient than the above method using disjoint outer polygons especially when the characteristic number K is large.

3.2 Horizontal Vertex Visibility Cells Graph

Using the horizontal vertex visibility partition of the outside of obstacles we can decompose the free space into $O(n)$ simple cells. Then we can represent the connectivity of these cells in terms of a graph and check the connectivity of this graph between the two cells containing the starting point p_S and the goal point p_G by doing a search on this connectivity graph, see Figure 4. There is a collision-free path between the starting and goal points if and only if there is a connected path between the starting and goal cells in this graph.

The hv-cells in Figure 4 have various configurations depending on the y-extreme vertices encountered in the construction. Assuming that there are no y-extreme vertices with the same y-coordinates, the hv-cells can have at most four adjacent hv-cells, at most two from the above and at most two from the below. Two adjacent hv-cells share a common horizontal edge which is one of the boundary edges of the hv-cells. When the path passes through an hv-cell, this path is entering this hv-cell through one of the horizontal edges and is exiting the hv-cell through the other horizontal edge. A connected path of hv-cells on the hv-cells-graph decides these entering and exiting edges on each hv-cell on the path. Depending on the local configurations of inter-connected adjacent hv-cells and the various heuristic strategies for path construction, we determine the entering and exiting points on the corresponding entering and exiting edges. The path construction is essentially reduced to constructing for each hv-cell a path segment connecting the entering and exiting points totally within the hv-cell. By adding intermediate pass points and the corresponding horizontal edges in the hv-cells we may assume the y-coordinate changes monotonically along the path segment within an hv-cell. We interpolate the horizontal ratios of the entering and exiting points on the corresponding edges. In Figure 5, α_1 and β_1 (with $\alpha_1+\alpha_2 = 1$ and $\beta_1+\beta_2 = 1$) are the entering and exiting horizontal ratios respectively. We may parameterize the constructed path segment so that the intermediate y-coordinates change linearly from the entering to the exiting y-coordinates. The corresponding intermediate horizontal ratio may be a linear interpolation $\gamma_1(t) = \alpha_1 \cdot (1-t) + \beta_1 \cdot t$ which is parameterized by t, $0 \le t \le 1$. Thus, at an intermediate y-level $y(t)$, the corresponding horizontal line is intersected with the left and right walls of the hv-cell, and with the corresponding left and right intersection points $p_l(t) = (x_l(t), y(t))$ and $p_r(t) = (x_r(t), y(t))$ we generate the path point $p(t) = (x(t), y(t))$ at time t, where $x(t) = (1 - \gamma_1(t)) \cdot x_l(t) + \gamma_1(t) \cdot x_r(t)$.

3.3 Motions with Translations and Rotations

We consider the gross motion planning of the robots moving with both translational and rotational degrees of freedom while avoiding collisions with obstacles in a plane. For the polygonal case, Avnaim, Boissonnat, and Faverjon [1] recently presented algorithms to generate the *C-space* obstacles for polygons moving with translations and rotations amidst polygonal obstacles. They construct the boundaries of three dimensional *C-space* obstacles in $O(m^3n^3\log(mn))$ time and the connectivity graph of volumetric prism cell decomposition of collision-free space in $O(m^6n^6\alpha(mn)\log(mn))$ time, where m and n are the numbers of edges in the moving polygon and the obstacle polygons respectively and $\alpha(mn)$ is the Ackermann's Inverse function which is an almost linear function of mn. However, it is not easy to extend this algorithm to the curved case, since the construction heavily depends on the discrete properties of line segments. It is still an important open problem to generate the exact *C-space* obstacle boundaries for planar curved moving objects and obstacles with both translational and rotational degrees of freedom. Further, the time complexity of this algorithm $O(m^6n^6\alpha(mn)\log(mn))$ for volumetric representation with prism cell decomposition of collision-free space is also quite pessimistic. Because of these difficulties, in the meantime we decided to use the conventional slicing method [16,17].

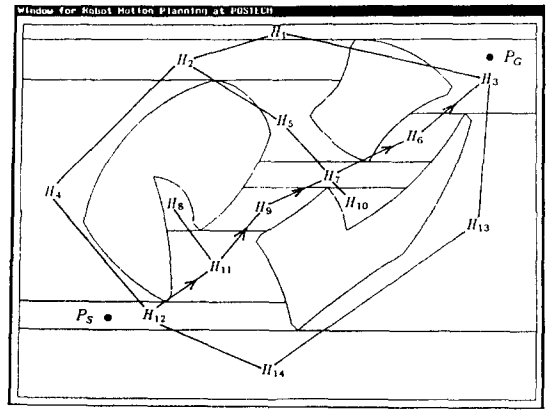


Figure 4: Connectivity Graph of Horizontal Visibility Cells

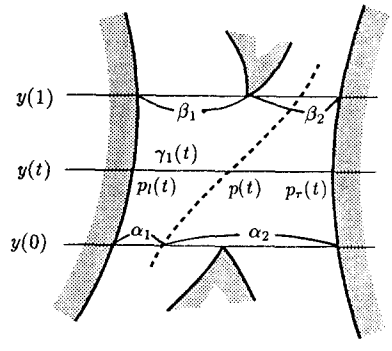


Figure 5: Path Construction

We divide the overall orientations into a certain finite numer of slices $[\theta_1, \theta_2], [\theta_2, \theta_3], \ldots, [\theta_{n-1}, \theta_n]$, where $\theta_1 = 0$ and $\theta_n = 2\pi$. At each orientation θ_i $(i = 1, \ldots, n)$, we construct the *C-space* obstacle CO_{θ_i} for the object moving with a fixed orientation at angle θ_i. We then compute the purely rotational sweep volume $A_{[\theta_i, \theta_{i+1}]}$ of the moving object over the orientation range $[\theta_i, \theta_{i+1}]$ $(i = 1, \ldots, n-1)$. The *C-space* obstacle $CO_{[\theta_i, \theta_{i+1}]}$ for

$A_{[\theta_i,\theta_{i+1}]}$ as a moving object is the set of all the positions where the original moving object can rotate without collisions with obstacles from orientation θ_i to θ_{i+1} or from θ_{i+1} to θ_i. Let FS denote the free space which is the complement of *C-space* obstacle CO. Figure 6(a) (resp. 6(b) and 6(c)) shows the hv-cells decomposition of FS_0 (resp. $FS_{\pi/6}$ and $FS_{[0,\pi/6]}$). Each connected component $S_{[\theta_i,\theta_{i+1}]}$ of $FS_{[\theta_i,\theta_{i+1}]}$ is totally contained in a connected component S_{θ_i} (resp. $S_{\theta_{i+1}}$) of the free space FS_{θ_i} (res. $FS_{\theta_{i+1}}$). For each $S_{[\theta_i,\theta_{i+1}]}$ and the corresponding S_{θ_i} and $S_{\theta_{i+1}}$, we have $S_{[\theta_i,\theta_{i+1}]} \in S_{\theta_i} \cap S_{\theta_{i+1}}$ and S_{θ_i} and $S_{\theta_{i+1}}$ are connected through any point of $S_{[\theta_i,\theta_{i+1}]}$. The object can move with translations and rotations as follows. The moving object with its reference point in S_{θ_i} (resp. $S_{\theta_{i+1}}$) first translates into a position in $S_{[\theta_i,\theta_{i+1}]}$, rotates by an angle $\theta_{i+1} - \theta_i$ (resp. $\theta_i - \theta_{i+1}$) without collisions with obstacles, and then translates to a position in $S_{\theta_{i+1}}$ (resp. S_{θ_i}). Any point in the connected component $S_{[\theta_i,\theta_{i+1}]}$ would be sufficient to connect the connected components S_{θ_i} and $S_{\theta_{i+1}}$ in different orientation layers. Though selecting one point from each connected component $S_{[\theta_i,\theta_{i+1}]}$ would be sufficient for the completeness of motion planning with slicing method, for a better performance we select one point $p_{[\theta_i,\theta_{i+1}]}$ from each hv-cell of $FS_{[\theta_i,\theta_{i+1}]}$ and connect the corresponding hv-cells H_{θ_i} and $H_{\theta_{i+1}}$ containing $p_{[\theta_i,\theta_{i+1}]}$ in the connectivity graph of the layered hv-cells-graphs. The motion of object from H_{θ_i} to $H_{\theta_{i+1}}$ and vice versa can be done in a similar way as with S_{θ_i} and $S_{\theta_{i+1}}$. Figure 6(d)-(e) show robot motions with translations and rotations.

4 Conclusion

We presented various algorithms for collision-avoidance robot motion planning with planar geometric models. Previous computational geometry algorithms for planar geometric models are assumed in this paper. These include the algorithms to compute

1. the *C-space* obstacle boundaries,
2. the rotational sweep volumes,
3. the convex hulls, and
4. various decompositions of planar geometric models.

We suggested motion planning algorithms using

1. the simple characteristic polygons,
2. the outer simple polygons,
3. the horizontal vertex visibility cells decompositions, and
4. the connectivity graph of inter-connected layered hv-cells-graph slices.

Some of these motion planning algorithms are implemented on SUN4/330GX Sparc Station using Common Lisp for the simple planar geometric models bounded by circular arcs and line segments. Further implementations are currently going on to deal with planar geometric models bounded by general planar algebraic curve segments.

References

[1] Avnaim, F., Boissonnat, J.D., and Faverjon, B., (1988), "A Practical Exact Motion Planning Algorithm for Polygonal Objects Amidst Polygonal Obstacles," *Proc. 1988 IEEE Int'l Conf. on Robotics and Automation*, pp. 1656–1661.

[2] Bajaj, C., and Kim, M.-S., (1988), "Generation of Configuration Space Obstacles: The Case of a Moving Sphere," *IEEE J. of Robotics and Automation*, Vol. 4, No. 1, pp. 94–99.

[3] Bajaj, C., and Kim, M.-S., (1989), "Generation of Configuration Space Obstacles: The Case of Moving Algebraic Curves," *Algorithmica*, Vol. 4, No. 2, pp. 157–172.

[4] Bajaj, C., and Kim, M.-S., (1990), "Generation of Configuration Space Obstacles: The Case of Moving Algebraic Surfaces," *The Int'l J. of Robotics Research*, Vol. 9, No. 1, pp. 92–112.

[5] Bajaj, C., and Kim, M.-S., (1990), "Convex Hulls of Objects Bounded by Algebraic Curves," to appear in *Algorithmica.*

[6] Bajaj, C., and Kim, M.-S., (1988), "Algorithms for Planar Geometric Models," *Proc. of the 15th International Colloquium on Automata, Languages and Programming (ICALP 88), Tampere, Finland, Lecture Notes in Computer Science*, Springer-Verlag, pp. 67–81.

[7] Bajaj, C., and Kim, M.-S., (1988), "Decompositions of Objects Bounded by Algebraic Curves," Computer Science Technical Report CSD-TR-677, Purdue University.

[8] Canny, J., (1988), *The Complexity of Robot Motion Planning*, ACM Doctoral Dissertation Series, MIT Press, Cambridge, Mass.

[9] Dobkin, D.P., Souvaine, D.L., and Van Wyk, C.J., (1988), "Decomposition and Intersection of Simple Splinegons," *Algorithmica*, Vol. 3, pp. 473–486.

[10] Dobkin, D.P., and Souvaine, D.L., (1990), *Computational Geometry in a Curved World, Algorithmica*, Vol. 5, No. 3, pp. 421–457.

[11] Graham, R., and Yao, F., (1983), "Finding the Convex Hull of a Simple Polygon," *Journal of Algorithms*, Vol. 4, pp. 324–331.

[12] Kim, M.-S., and Moon, S.-R., (1990) "Rotational Sweep Volumes of Objects Bounded by Algebraic Curves," *Proc. 1990 IEEE Int'l Conf. on Robotics and Automation*, Cincinnati, Ohio, pp. 311-316.

[13] Kim, M.-S., and Lee, I.-K., (1990) "Gaussian Approximations of Objects Bounded by Algebraic Curves," *Proc. 1990 IEEE Int'l Conf. on Robotics and Automation*, Cincinnati, Ohio, pp. 317–322.

[14] Lee, D.T., (1983), "On Finding the Convex Hull of a Simple Polygon," *International Journal of Computer and Information Sciences*, Vol. 12, No. 2, pp. 87–98.

[15] Lee, D.T., and Preparata, F.P., (1984), "Computational Geometry – A Survey," *IEEE Transactions on Computers*, Vol. C-33, No. 12, December 1984, pp. 872–1101.

[16] Lozano-Pérez, T., (1983), "Spatial Planning: A Configuration Space Approach," *IEEE Trans. on Computers*, Vol.C-32, pp. 108–120.

[17] Lozano-Pérez, T., and Wesley, M.A., (1979), "An algorithm for planning collision free paths among polyhedral obstacles," *Communications of the ACM*, Vol. 22, pp. 560–570.

[18] Ó'Dúnlaing, C., Sharir, M., and Yap, C.K., (1983), "Retraction: A New Approach to Motion-Planning," *Proc. of the 15th Symp. on the Theory of Computing*, Boston, pp. 207–220.

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

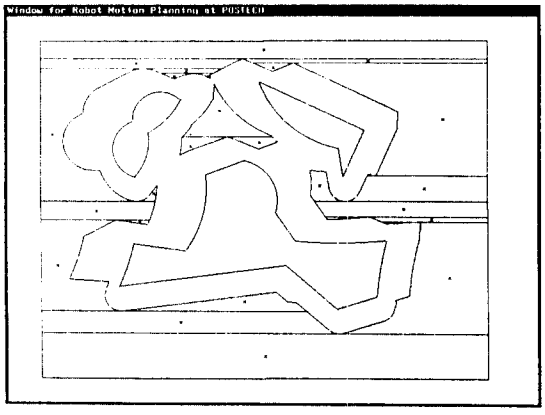
[20] Schäffer, A., and Van Wyk C., (1987), "Convex Hulls of Piecewise-Smooth Jordan Curves," *Journal of Algorithms*, Vol. 8, No. 1, pp. 66–94.

[21] Schwartz, J.T. and Sharir, M., (1983), "On the Piano Movers Problem: II, General Techniques for Computing Topological Properties of Real Algebraic Manifolds," *Advances in Applied Mathematics* Vol. 4, pp. 298–351.

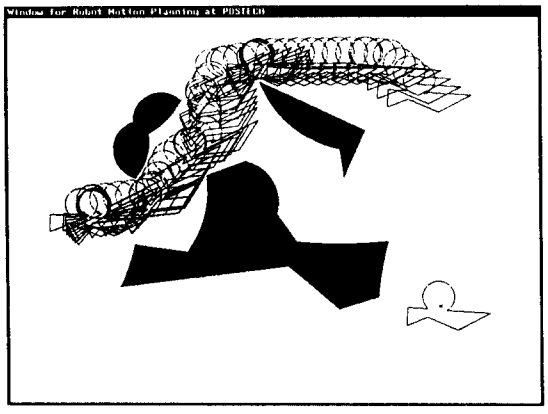
[22] Tarjan, R.E., and Van Wyk, C.J., (1988), "An $O(n \log\log n)$-Time Algorithm for Triangulating Simple Polygons", *SIAM Journal on Computing*, Vol. 17, No. 1, pp. 143–178.

[23] Walker, R., (1978), "Algebraic Curves," Springer Verlag, New York.

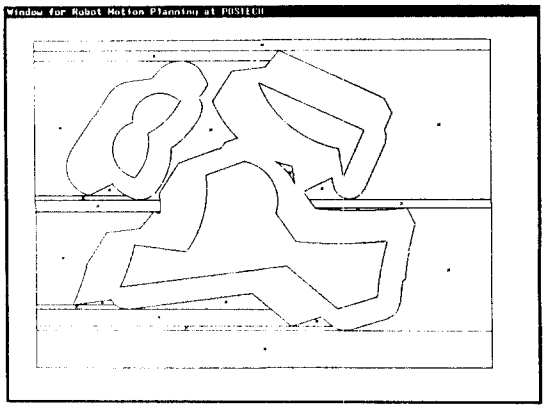
[24] Yap, C.K., (1987), "An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments," *Discrete and Computational Geometry*, Vol. 2, No. 4, 1987, pp. 365–393.



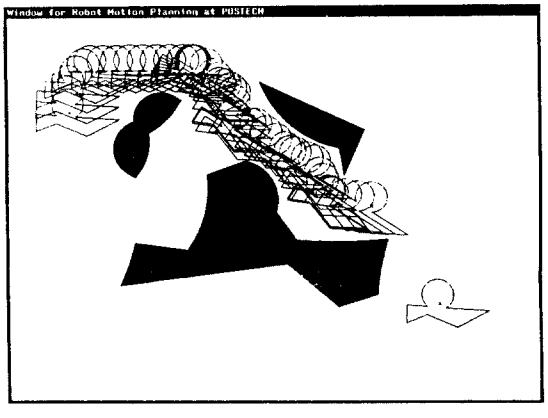
(a)



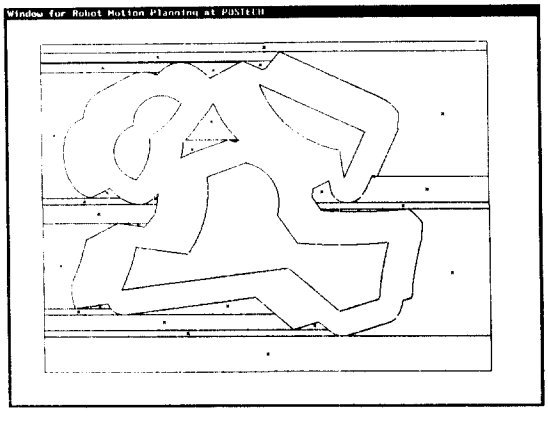
(d)



(b)



(e)



(c)

Figure 6: (a) FS_0. (b) $FS_{\pi/6}$. (c) $FS_{[0,\pi/6]}$. (d)-(e) Motions with Translations and Rotations.