CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible

If pages are missing, contact the university which granted the degree

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

Algorithmic Motion Planning in Robotics

Geetha Ramanathan

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

April 1985

ABSTRACT

Algorithmic Motion Planning
in Robotics

Geetha Ramanathan

Algorithms for collision free planar motion of one or more disks amidst polygonal obstacles are given. For k circular disks the time complexity of our algorithm is shown to be $O(n^k)$ where n is the number of edges composing the walls. This algorithm is conjectured to be optimal.

Acknowledgements

I would like to acknowledge the tremendous support I received from Dr. V. S. Alagar during the development of this thesis. His guidance was invaluable and I am grateful to have had the opportunity to work with him. I also acknowledge considerable financial assistance from Dr. V. S. Alagar.

Table of Contents

Chapter 1

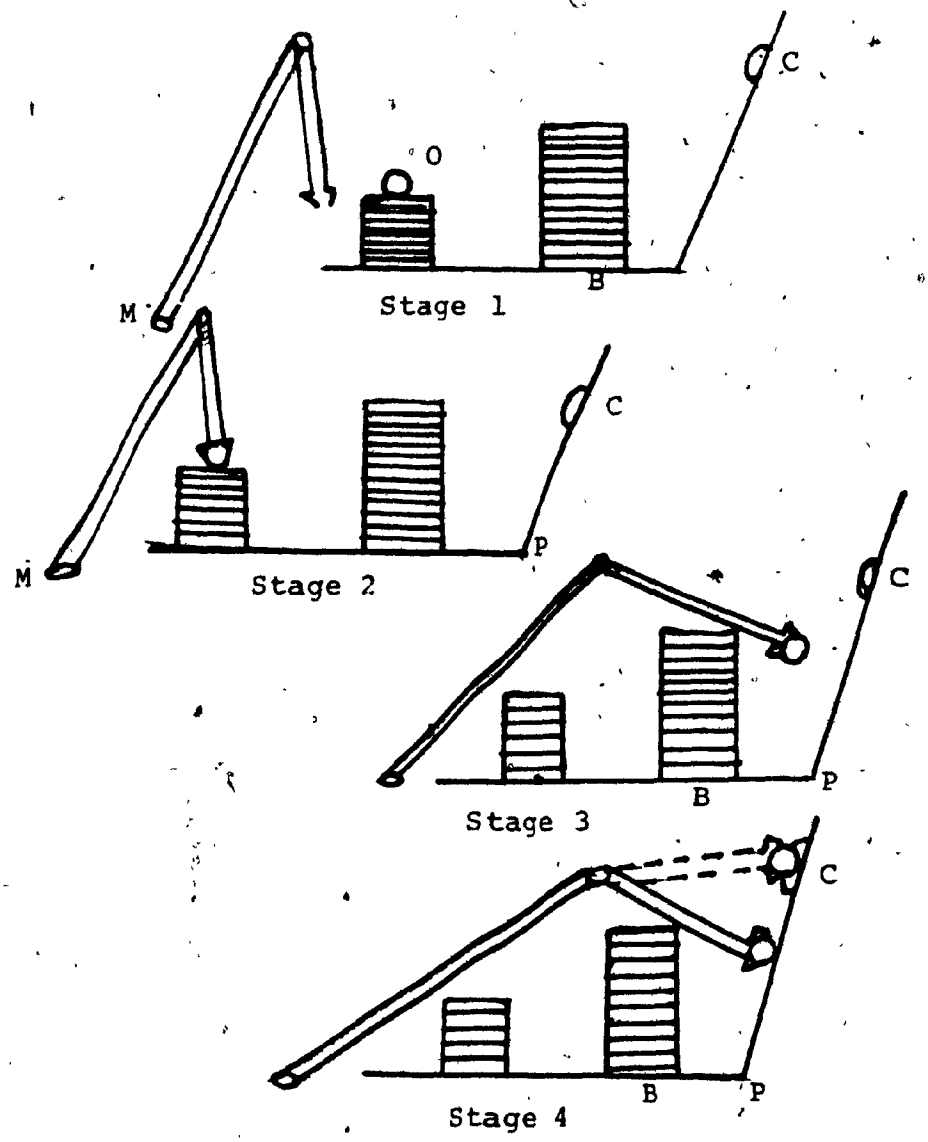
Introduction

1.1 Robotic manipulation problem

Robotics is an interdisciplinary field of study comprising of diverse topics such as vision, motion planning and control, actuation and manipulator design. Even within one topic such as manipulator planning and control, we can identify major issues such as dynamics, feedback control, trajectory planning, compliance and task planning. As robots are demanded to perform sophisticated tasks, a great deal of advances in each one of the above areas is necessary to support the expected level of sophistication. A sound and complete specification of the tasks is an essential prerequisite to design the geometric structure of a robot and develop its versatile mechanisms.

A basic problem common to a range of tasks in robotics is planning motions and then controlling the motion to suit the environment. The motions of a robot are known as trajectories consisting of a sequence of points with velocities associated with these points. For a robotic manipulator consisting of rigidly linked arms, the kinematics of motion planning requires well-known techniques in Lagrangian and Newtonian dynamics. We are not concerned with the mechanical and control aspects of motion planning. Our interest is in basic task planning, collision avoidance and path finding within a robotic manipulation system. We illustrate these aspects of robotic manipulation through the

configuration illustrated in Figure 1 and the range of tasks described below :



1. Lift the object O and place it on P.

2. Move the object O along P without hitting obstacle B and then place it in the hole C.

For that purpose, we shall consider a simple two link manipulator M where both the joints are revolute. The links

can only revolve around the joints. In the first step, the end-effector has to reach for the object and grasp the object firmly without hitting it. This is shown in stage 2. A typical robot operation begins with such grasping. The rest of the operations are influenced by choices made during the grasping. It is important to devise collision free grasp motions and these are independent of obstacle avoidance methods.

Next, the positioning system must execute a trajectory so that the obstacle B is avoided and the object O comes into contact with plane P, past B. This path must be planned accurately by the manipulator so that the task is completed without collision. Moreover, the contact of O with P must be established without hitting the plane. This is shown in stages 3 and 4. Finally, the control system must take into account the configuration of the plane and move O into the hole so that the contact with the plane P is maintained throughout the motion upto the hole.

At the end of this motion, the object O must be slid into the hole. This last two stages of motion as well as the motion in stage 2 are known as compliant motion; that is, the motion planned by the manipulator when it is in contact with an external surface. In stage 3, there are two different aspects of manipulator actions. The first one is called trajectory planning which converts the desired motion, given the initial and final points, to a sequence of intermediate configurations of the arm. The second aspect

is avoiding collision with the obstacle B for which the manipulator should compute the set of free positions so that the motion can be guided through the set of free positions. In other words, the task planner must determine manipulator paths which avoid collisions.

Collision avoidance is an important basic issue in all types of motion planning. Given a model of the manipulator, the geometry of obstacles in the work space and the general topology of the work space, finding a collision free path for a moving solid among other obstacles is a fundamental problem in robotic manipulation.

1.2 Task planning : Algorithmic aspects

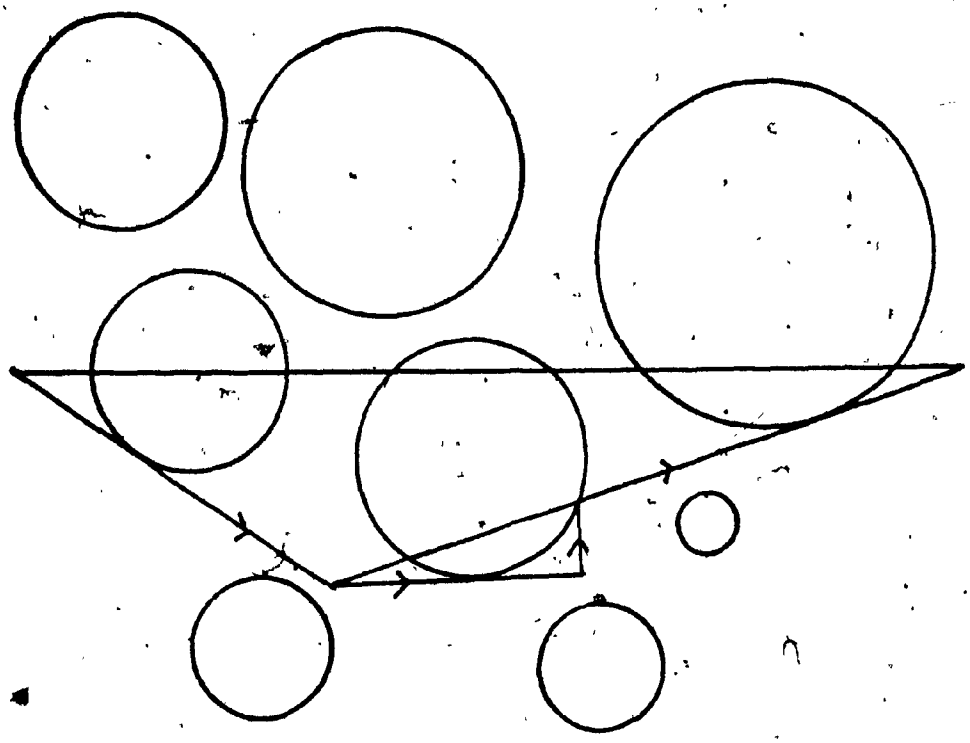
In order to automate a robot application, it is important to have an accurate specification of the tasks, an exhaustive listing of the tasks and a transformation or an algorithm to transform the task level specification into manipulator level specifications. We assume that geometric descriptions of all objects and the robot and the topology of the environment are specified accurately.

There are three different algorithmic approaches to collision avoidance in task planning. These are grouped into the following :

1. Hypothesize and test methods [10,15]
2. Penalty function methods [5]
3. Computing explicit free space [2,3,11,12,19,20,21,23]

Hypothesize for obstacle avoidance test is the earliest proposal and is largely heuristic. An initial path from the

initial to final configuration is assumed and is then tested for collisions. If collisions are detected the information gained locally during the collision test with an object is



used again to hypothesize and test a new path. See figure above.

The main advantage of this approach is its simplicity because the entire set of operation is the repetition of one basic operation. This method has severe drawbacks. The information gathered during a collision is purely local and will in general provide little guidance on how a path may have to be modified to avoid a collision. Moreover, it is impossible to deal with multiple collisions; for the complexity of path modification subsumes the overall

complexity of path finding. Even if we are lucky enough to find a path there is no guarantee that this is the shortest path. During the formative stages of our research work, we attempted similar approaches but abandoned them because an explosive number of situations were to be examined in a densely cluttered environment.

The second approach based on penalty functions computes a path based upon the local minimum of a penalty function. The penalty function itself is computed by adding some preassigned penalties for obstacles and adding a term for deviating from the best possible path. This method, lends itself to analytic techniques and provides attractive solutions to simplistic models. However, there is no obvious way to assign penalties to obstacles. One approach to get around this difficulty is to define the potential of the robot in a field and the obstacles be replaced by repelling forces. The motion is determined by solving a set of kinematic equations. The main drawback of such an approach is that the method relies only on local information and hence a considerable amount of backtracking is necessary to choose, test and try alternate configurations. This suggests that more global information on the scattered set of obstacles be gathered first to determine an error free and collision free path for the object.

The third approach is to analyze the entire work space of the robot and compute the set of free positions so that a path can be defined as a sequence of free positions. The

methods used to compute this free space and its representation to determine the existence of a path vary widely. Even for simplistic robotic models and obstacles, the complexity of such computations remains high. However, the main advantage in this approach is the certainty with which a path can be found when one exists. Moreover, it is entirely feasible to search and find best paths, rather than just find the first safe path. Efficient obstacle avoidance algorithms for robots of general geometric structure are not known; however, a theoretical formulation has been recently given by Schwartz and Sharir [20].

1.3 An overview of thesis

This thesis discusses efficient algorithms for computing the free space for collision free motion of simplified robotic models in a plane. Details of their implementation are not given in the thesis although it is straightforward.

As mentioned above and to be discussed further in chapter 2, the cost of computing all the connected components of free space is expensive. However, it is extremely important to capture such a global information for accurately determining a collision free motion. With this spirit in mind, the thesis motivates in chapter 3 through known complexity results on motion planning, the necessity to investigate simple models.

Chapter 4 outlines methods to determine collision free motion of a single disk (a simple robot) among polygonal or

circular obstacles.

We consider in Chapter 5, coordinated motion of two independent disks moving amidst polygonal barriers. A generalization for k (> 2) disks is given in chapter 5.

Chapter 7 presents a summary of the methods discussed in the thesis, discusses optimality of our algorithms and suggests generalization of our approach to objects of other shape.

Chapter 2

Complexity of motion planning in robotics

2.1 Introduction to complexity issues

There is considerable progress in the development of mechanical devices autonomously controlled by micro and mini computers and there is a steep increase in their computational power. Mechanical devices for robotic manipulation and computer controlled arms have been discussed by Paul [15]. Udupa [24] and Lozano-Perez and Wesley [12], have discussed general methods for collision detection and avoidance in computer controlled robotic manipulators. However, none of these algorithms are guaranteed to run in polynomial time. Moreover, they are only approximate algorithms and this led Reif [18] to examine the computational complexity issues in robotics.

Reif [18] considered abstractions of collision avoidance problem called the classical movers' problem and the generalized movers' problem. The classical movers' problem was posed as a decision problem in the following way :

Instance : Given a set of obstacles O, two distinguished positions I (initial) and F (final) fixed in Euclidean n-space and B, a rigid body (a robot).

Question : Can B be moved by a sequence of translations and rotations from I to F without colliding with any obstacle in the set O ?.

In general, robot's motion must be a sequence of

continuous motions. The set of obstacles in O are polyhedrons. In a discrete version of the above problem, the polyhedrons can be replaced by a system of linear inequalities (within a fixed accuracy) and the motion itself when it exists will be given by a sequence of points and the motion is continuous between every successive pairs of points. Reif [18], also considered a generalized version of the mover's problem in which the polyhedra are linked together at some distinguished vertices. A particular instance of this being, a robot arm with multiple joints. We will give the main result of Reif after briefly mentioning some definitions in computational complexity.

In the study of computational complexity, it is usual to denote by P, the class of problems for which polynomial time algorithms are known and by NP, the class of all problems which can be solved by non-deterministic polynomial time algorithms. It is strongly believed that $P \neq NP$. All problems in NP - P (assuming that the conjecture $P \neq NP$ is true) are intractable, and are known as NP-complete problems. In other words, if $P \neq NP$ then an NP-complete problem is in NP - P.

In solving a problem, the amount of storage or computer memory required by the computation, is known as space complexity. A problem solvable in polynomial time is also solvable in polynomial space. However, it is still unresolved whether there exists problems solvable in polynomial space that cannot be solved in polynomial time.

This conjecture is quite plausible, [Garey and Johnson] since all problems in NP and all problems in # P (the class of hard counting problems) can be solved in polynomial space. Hence, it appears that there are problems that can be solved in polynomial space which are harder than the problems in NP - P and # P. PSPACE is the class of problems solvable by polynomial space and PSPACE-complete contains the hardest problems in the sense that their solutions are much harder than the problems in #P and NP-complete classes. Hence, if a problem is shown to be PSPACE-complete then it is a strong indication that it is more intractable than an NP-complete problem.

In the light of these notions of complexity, we summarize below the results due to Reif [18], Hopcroft, Schwartz, Sharir [5], and Hopcroft, Joseph and Whitesides [7].

2.2 Complexity results on movers' problem

Result 1 [Reif]

The generalized movers' problem is PSPACE hard.

Result 2 [Reif]

There exists polynomial time algorithms for solving the classical movers' problem in two and three dimensions.

In a classical movers' problem, the object to be moved is a rigid object and the region can be described by inequalities. In the generalized movers' problem, the object X has joints and hence is non-rigid. Reif's result asserts that the motion of arbitrarily hinged objects even

in 3-dimensions is PSPACE complete.

Result 3 [Hopcroft, Joseph, Whitesides]

Suppose n, k, I_1, I_2, ... , I_n are positive integers. Consider a carpenter's ruler consisting of n links. L_1, L_2 , ... , L_n that are hinged together at their end points with length of $L_j = I_j$. The problem of determining whether this ruler can be folded (each pair of consecutive links forming either 0 or 180 angle at the joints) so that the folded length is utmost k, is NP-complete.

It is fairly easy to see that by a reduction from PARTITION, the well-known NP-complete problem [4], the above result can be proved to be NP-complete. Moreover, the ruler folding is analogous to moving a hinged arm in a bounded 2-dimensional region. Hence, it turns out that it is NP-hard to decide whether the end of an arm with n links of arbitrary lengths (with one end fixed) can be moved from one position to another while staying within a two dimensional

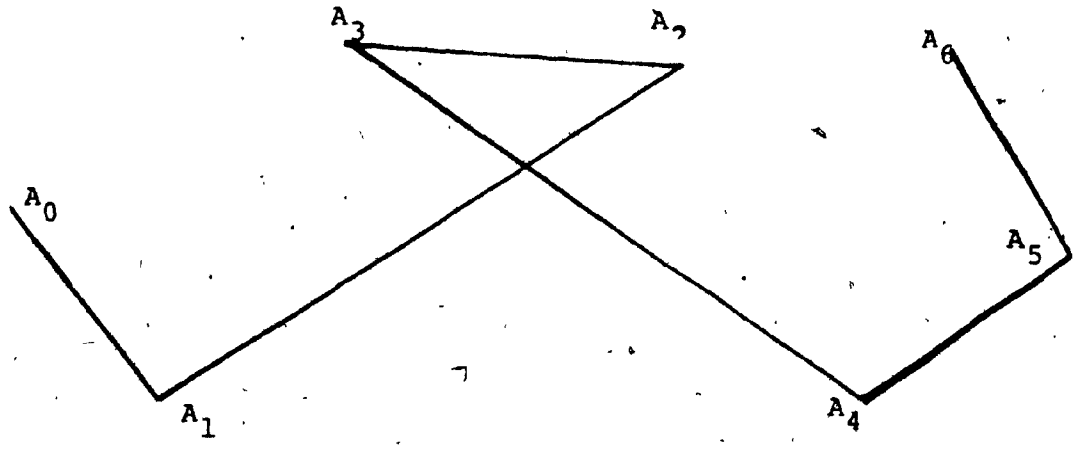


Figure 2.1

region. See figure.

Result 4 [Hopcroft, Joseph, Whitesides]

There is a polynomial time algorithm for moving the arm with n links (with one end fixed) from one configuration to another inside a circular region.

This result does not apply when the arm is constrained to move to avoid specified obstacles during its motion. It is not yet known whether the problem of determining such motion is polynomial or NP-hard.

The above two results show particular instances of generalized movers' problem in two dimensions. One of them turns out to be NP-complete and the other is polynomial. One can inquire about similar results for objects that are not hinged but are free to move independently. The following is the first known instance of a "hard" problem for the motion of independent objects in two dimensions.

Result 5 [Hopcroft, Schwartz, Sharir]

The coordinated motion planning for arbitrarily many rectangles in two dimensions amidst polygonal obstacles is PSPACE hard.

This is a significant result because the motion planning problem involving moving systems whose geometry is simple to describe is still a hard problem to solve. This leads us to examine simple geometric moving models in two dimensions and investigate the complexity of their motion planning. When the number of degrees of freedom of the moving bodies is held fixed and the entire system of objects

and obstacles are described algebraically, it is shown by Schwartz and Sharir [20] that the motion planning problem can be solved in polynomial time.

Chapter 3

Algorithms for collision free motion planning : A brief review

3.1 Introduction

The class of obstacle avoidance algorithms which construct explicit subset configurations for objects that are free of collisions is known as free space computation algorithms. Once the free space is computed through a global analysis of the region, the subsets of free space can be represented in a suitable manner and the path finding problem is then reduced to a search in this lower dimensional free space. A connectivity graph is then defined and the existence of a collision free motion is related to the existence of a path in this graph.

Udupa [24], is one of the first to develop this free space method; however, his work is special to Stanford arm. Lozano-Perez and Wesley [12] is the first general algorithm applied to derive the motion of a polyhydron moving in an arbitrary topology of polyhedral obstacles in a bounded space. We describe this method first. Next we review two methods related to circular disks moving among polygonal barriers. One of these applies to the motion of a single disk and the other discusses coordinated motion of two disks. These two specific algorithms are chosen because they are the best known algorithms and our algorithms to be discussed in chapters 4, 5 and 6 achieve both generality and efficiency over these methods.

For the sake of simplicity, consider a single point moving among polygonal obstacles in a bounded two dimensional region. In this case, the forbidden regions are the interior of the polygons which the moving point should avoid in its continuous motion. Now a safe path is found by defining a graph whose vertices include the vertices of the polygonal regions and finding a path through this graph connecting vertices of the forbidden regions.

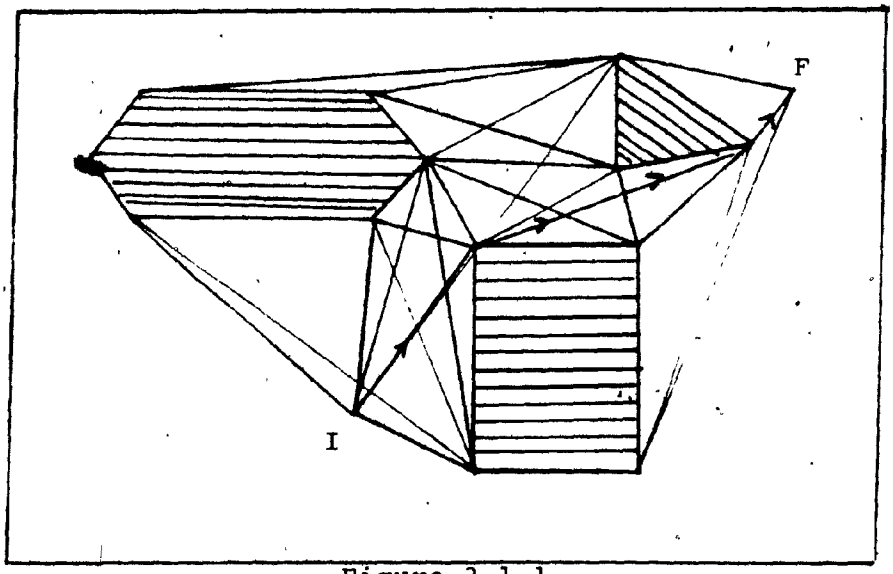


Figure 3.1.1

The example in figure 3.1.1 shows the forbidden polygonal regions, the graph and the collision free path for a point. Note that there is an edge between two vertices in the graph if and only if each is visible from the other. That is, the edge does not intersect any forbidden region. The graph is known as the visibility graph of the given configuration and the shortest path in the graph from the initial point to the destination is the best possible solution to this problem.

The above method provides a first approximate solution

to problems when the object size is insignificant to the size of the obstacles. However, it is important to take into account the dimensions of the robot to obtain an exact solution. For example, consider the object to be a circle of radius r. Since the visibility graph (VGRAPH) algorithm requires that the moving object be a point, we should transform the given situation to one wherein the object is a single point with a new set of obstacles. These new obstacles are the forbidden regions for the point.

The new set of obstacles, also known as the grown-up obstacle set, is computed by moving each vertex away from the obstacle to be at a distance at least r from the sides. See figure 3.1.2 and observe that the collision free path obtained is different from figure 3.1.1. Note that the grown-up obstacles are only approximations to forbidden regions.

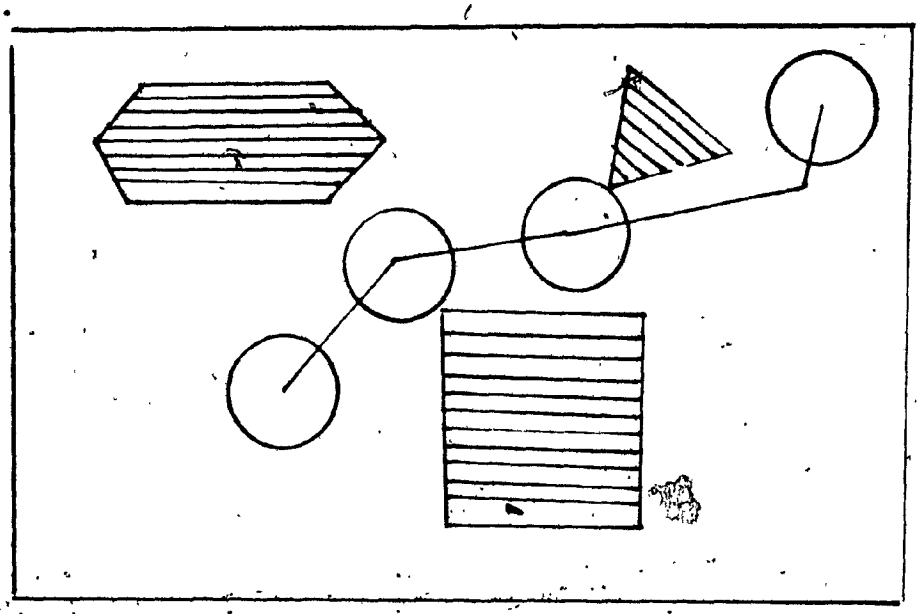


Figure 3.1.2

The problem of computing the grown-up obstacles is

slightly more complicated for polygonal objects, for it depends on the orientation and the choice of reference point of the object. In the case of a circle, the center of the circle is a reference point in the sense that the original problem is reduced to moving this reference point among the grown-up obstacles. When we consider a polygonal object, lack of symmetry suggests choice of any point on the boundary of the polygon as the reference point. Figure 3.1.3 and 3.1.4 show two choices of reference points with different orientations of the object and the corresponding grown-up obstacles. We explain below the method of growing up obstacles, taking into account the orientation of the object and a given reference point.

Each side of an obstacle is grown towards the set of free positions and this grown-up side is obtained as the locus of the reference point while keeping the object in contact with the obstacle and preserving the orientations. Because two objects should touch before they can collide, every position outside this traced out boundary must be a collision free position for the object with the given orientation fixed. However, when the orientation need to be changed at specified places, then a rotation of the object at those positions should be computed to produce the grown-up boundary. When two grown-up obstacles have a common interior then the set of vertices of the resulting forbidden region cannot contribute to edges in the VGRAPH. We remark that the process of growing up obstacles is

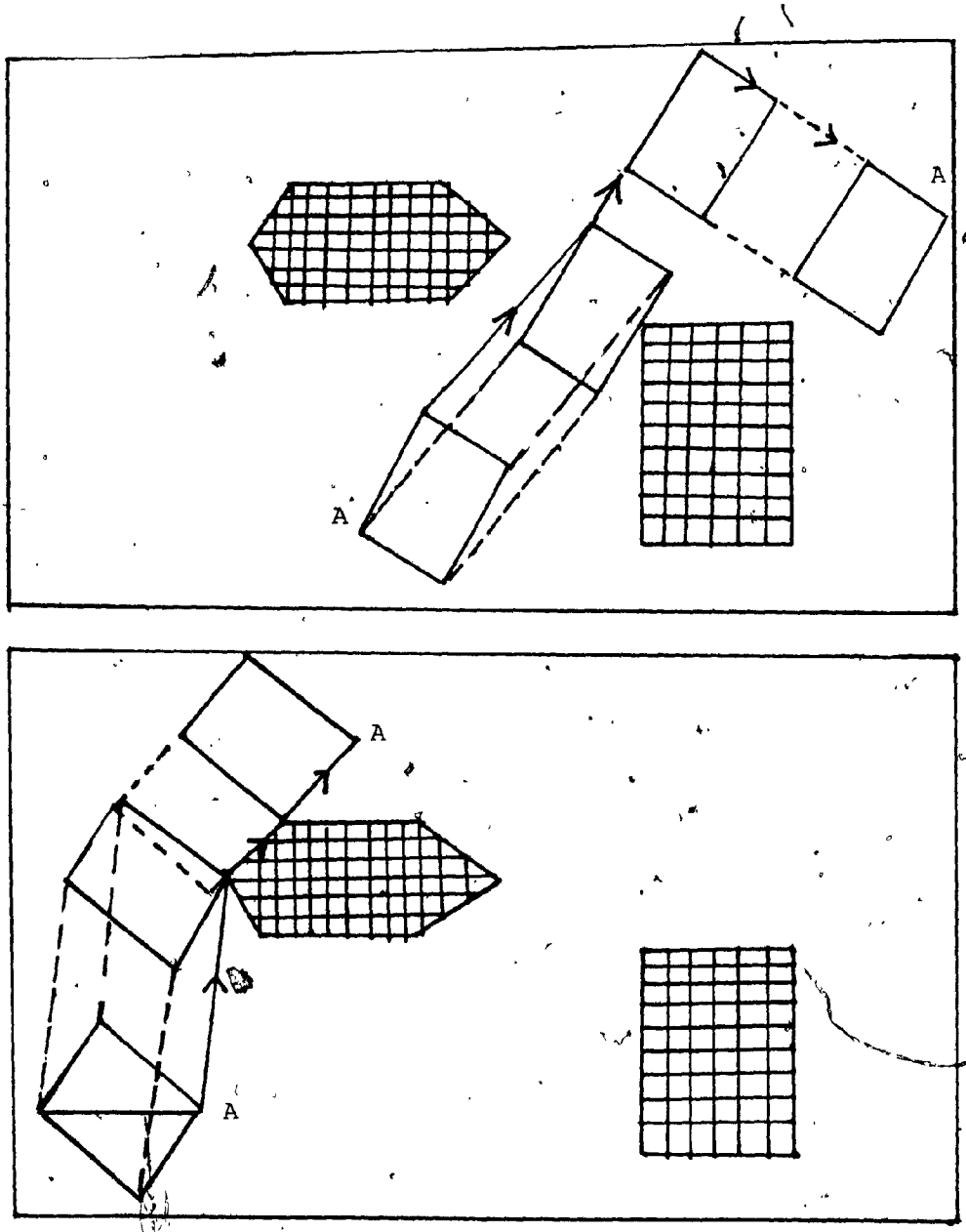


Figure 3.1.3

sensitive to the orientations.

Suppose the orientation of the object in its initial position is α and is different from the orientation β in its final position. If there is a position wherein the object can be moved with the orientation fixed at α and a rotation is applied to change the orientation from α to β at this point then the method can be applied with β fixed. The main

drawback of this approach is that the path in the graph may not exist although there might be a motion involving a continuous change of orientation. Hence, this is only an approximate algorithm.

3.2 Motion planning for a disk

A collision free motion planning method based on a continuous map called retraction has been given by O'Dunlaing, Sharir and Yap [14]. Informally, this method tries to move a body by trying to keep it equidistant from all the obstacles at all times during its motion. In practice, this may not be possible to achieve. So, a generalized Voronoi diagram is defined as the subset of the configuration space of free positions such that a placement at a Voronoi vertex would be nearest to two or more obstacles. When the object is a disk, the image map of the set of free positions onto the Voronoi diagram is continuous and is called retraction. For any placement X of the disk, this retraction map pushes the object away from the obstacle closest to the disk at X until the object reaches a placement on the Voronoi diagram.

For each point P_i from a given set of n points, a convex polygon C(i) called a Voronoi polygon can be associated with the property that P_i is the closest of the given points to any X in C(i). The collection of Voronoi polygons form Voronoi diagram. An example of a Voronoi diagram for points is given below.

Shamos and Hoey have given an O(n logn) algorithm to

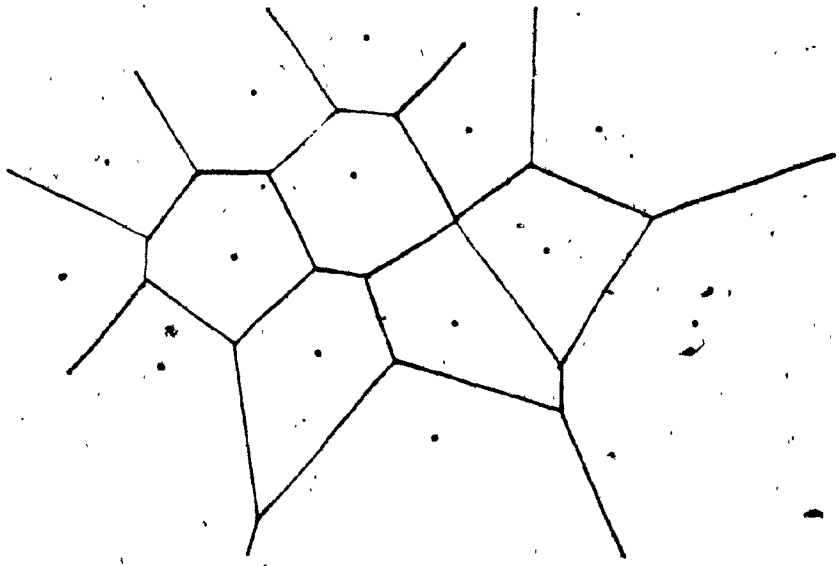


Figure 3.2.1

construct the Voronoi diagram of n points in the plane. This notion has been generalized to compute the Voronoi diagram of other geometrical figures. In particular, Kirkpatrick [8] has given an O(n log n) algorithm to compute the Voronoi diagram of n points and line segments. Figure 3.2.2 shows the Voronoi diagram for two closed line segments.

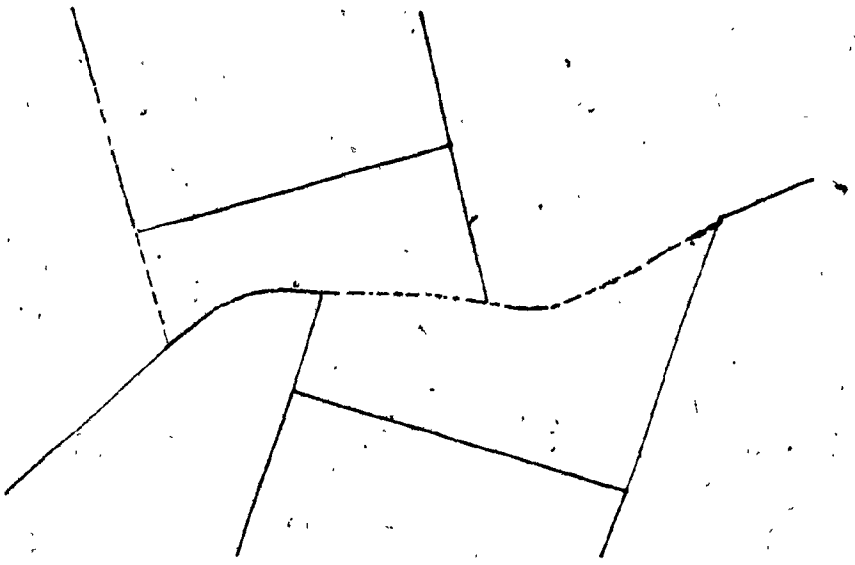


Figure 3.2.2

Let O be the set of polygonal obstacles. Given two points X and Y, let us denote the Euclidean distance between them as $E_d(x,y)$. If S is a non-empty set of points, the Hausdorff distance $d_H(x,S)$ is defined as :

$$\min \{ E_d(x,y) \ / \ y \text{ is in } S \}.$$

Because we consider a bounded region enclosed by a closed polygon with all its obstacles in its interior, the Hausdorff distance $d_H(x,O')$ is well-defined where O' also denotes the union of all the obstacles and the outer boundary. This distance $d_H(x,O')$ is called clearance(x) and this distance is attained at some point P on the boundary of some obstacle. For each point x, define near(x) to be the set of points in the boundaries of O' attaining the distance clearance(x). Now, the Voronoi diagram VOR(O'), for O', is the set

$$\{ x \ / \ \text{near}(x) \text{ contains more than one point} \}.$$

If r is the radius of the given disk, the set of free positions is identical to the set of points x for which clearance(x) $\geq$ r. Hence, VOR(O') is a subset of the set of positions for a given disk.

The retraction map M is defined as follows :
For any point x in VOR(O'), define M(x) = x. For any x not on the Voronoi diagram, near(x) must contain a unique element y. Now, join xy and consider the first point z where the half line from y to x meets the Voronoi diagram. Now, define M(x) = z. It can be shown that M is a continuous retraction of the set of points within the outer

boundary onto the Voronoi diagram. Moreover, the value of clearance(t) increases as t moves along the line joining x and M(x).

Suppose x and y are two free positions, then the following algorithm determines the path if there is one.

Step 1. Using Voronoi diagram construction given in [9] and the definition near(x), determine VOR(O'). In this diagram, the edges along the polygonal boundaries determine Voronoi cells. Each cell consists of points closest to a specific polygonal edge or a corner. Because there are O(n) edges overall, the minimum clearance along the edges can be computed in time O(n).

Step 2. For any two points x and y, determine M(x) and M(y). Note that a search of the Voronoi diagram can be done to find the cell containing x if it is not already on the diagram. Then, M(x) can be found by taking the point of intersection. Similarly, we can do for y. This step can be done in O(n) time.

Step 3. If M(x) and M(y) are on the same Voronoi edge then it is sufficient to determine the minimum clearance along that edge. If it is greater than r then there is a motion from M(x) to M(y). Otherwise there is no motion from M(x) to M(y). If M(x) and M(y) belong to two different Voronoi edges then we examine whether an end-point of the edge containing M(x) can be reached from an end-point of the edge containing M(y) while maintaining a minimum clearance of r. If the answer is no, then there is no motion. Otherwise,

there is a path between two Voronoi vertices.

Step 5. The existence of a path between M(x) and M(y) asserts the existence of a motion between x and y; because if there is a path from x to y then this path can be projected by the retraction map M to yield a path from M(x) to M(y) along the Voronoi boundary. Conversely, any path from M(x) to M(y) along the Voronoi boundary assures the existence of motion from x to y because the clearance function strictly increases along the line joining x and M(x).

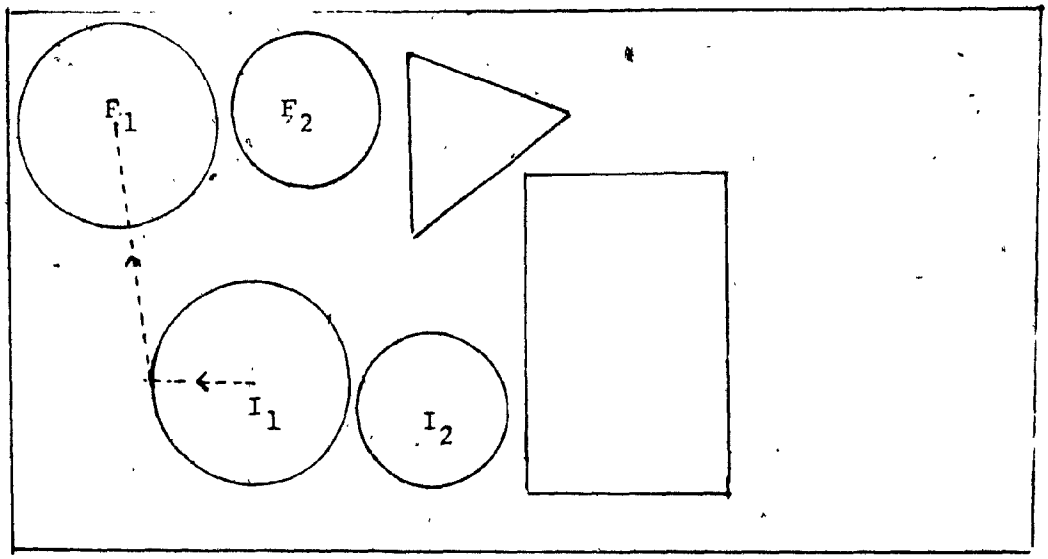
The Voronoi diagram can be constructed in time O(n logn). For points x and y, locating the Voronoi cells and computing their clearances can be done in linear time. The image points M(x) and M(y) can be obtained in linear time. The Voronoi graph can be searched to find the edges with minimal clearance in linear time. Thus, the overall cost of the algorithm is O(n logn).

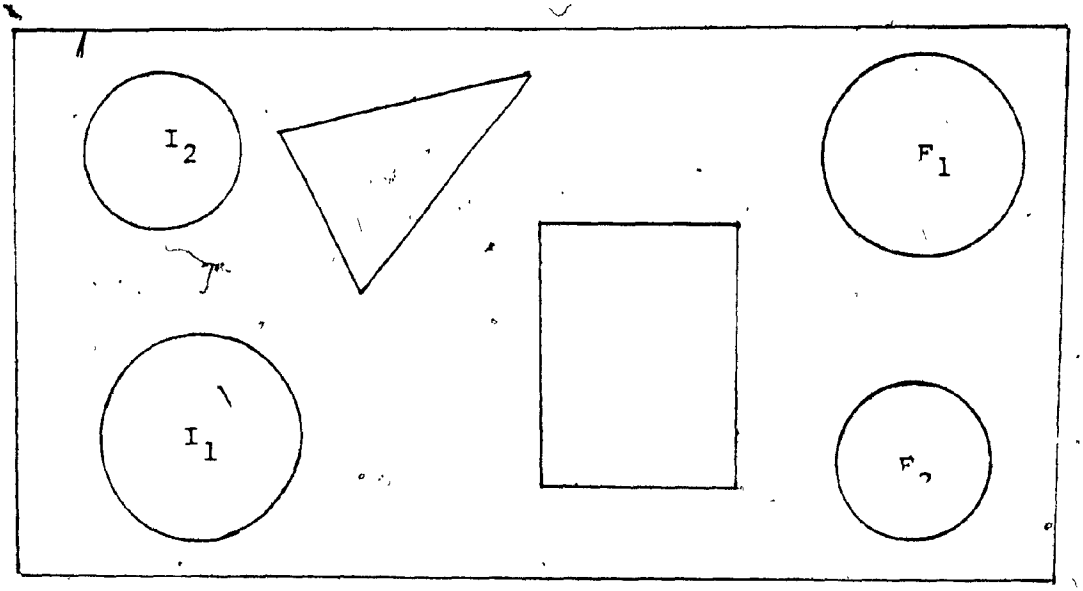
3.3 Coordinated motion of two disks

Schwartz and Sharir [19] gave the first exact algorithm for determining the existence of the coordinated motion of two disks among polygonal obstacles. The approach is based on studying the set of free positions for the disks and obtaining a decomposition of it into connected components. Since the two disks can move independently of each other without collision, the motion involves four degrees of freedom.

Figures 3.3.1 and 3.3.2 show two instances of a pair of

circles, initial and final configurations. In the first figure, a motion is shown by a dotted path. In the second figure, there is no motion.





Suppose D_1 and D_2 are two circular bodies with centers C_1 and C_2 and radii r_1, r_2 respectively. Assume $r_1 \geq r_2$ and the region R in which the disks are to be moved freely is a two dimensional open region bounded by a set of polygonal

walls. That is, R has a polygonal exterior boundary W_0 and encloses a number of polygonal shaped obstacles O_1, O_2, ... O_{p-1}. Let W_i denote the boundary of the obstacle O_i. A point X in R is called admissible for D_1 if when C_1 is placed at X, D_1 does not penetrate any wall. Clearly, any point X is admissible for D_1 if and only if X is at least at distance r_1 from every wall. A line segment at distance r_1 from a wall is called a r_1-displaced wall. The displacement of a wall is effected towards the set of admissible positions of a disk. A circular arc of radius r_1 from a convex corner of a polygonal wall is called a displaced corner. Note that, if the interior angle at a vertex of the outer polygon is obtuse then the vertex is convex; for an inner polygonal boundary, a vertex with acute interior angle is convex. The displaced walls and the displaced corners are formed for each obstacle with respect to each disk. The enlarged boundary of an obstacle is one whose edges are the displaced edges and displaced corners of that obstacle and whose vertices are the mutual intersections of their edges. Figure 3.3.3 explains these concepts for one disk.

The first step in their algorithm is to obtain the r_1-boundary and r_2-boundary of the boundaries W_i, $0 \leq i \leq p-1$. The collection of boundaries enlarged by the amount r_1 is called r_1-boundary. Similarly, r_2-boundary is defined. A point X is called critical if $\Gamma_{12}(X)$, the circle center X and radius $r_1 + r_2$, either passes through the point of intersection of two r_2-displaced walls (two r_2-displaced

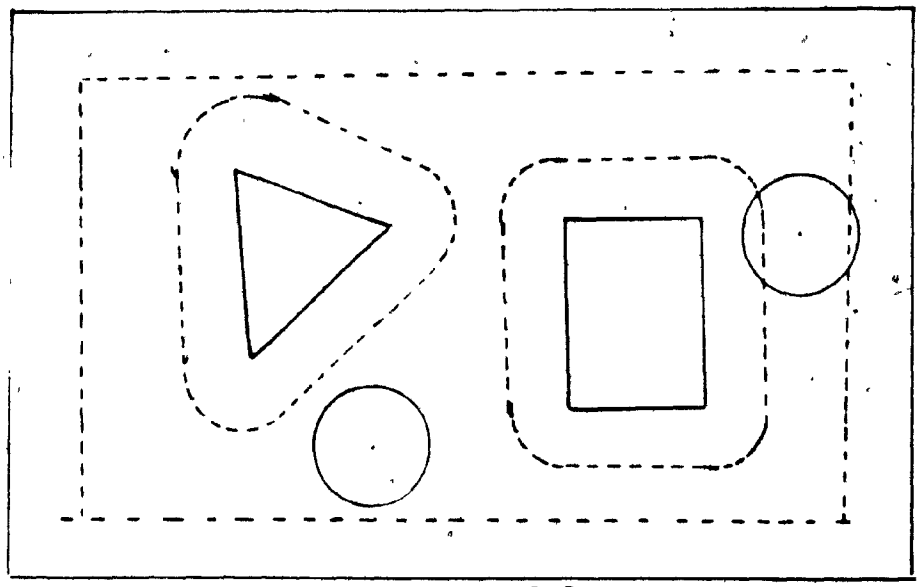


Figure 3.3.3

corners) or is tangent to a r_2-displaced wall (r_2-displaced corner). A critical curve is a collection of critical points and for a given set of critical points their exists several critical curves.

The set of critical points can be classified into three kinds of curves. These are.

1. straight line segments displaced by the amount $r_1 + r_2$ from r_2-boundary.

2. circular arcs at distance $r_1 + r_2$ from the r_2-displaced corner.

3. circular arcs which lie at distance $r_1 + r_2$ from a convex corner at which two r_2-displaced walls or r_2-displaced corners meet. These curves fall into two types.

Type 1. For each edge, the locus of all the points at distance $r_1 + 2r_2$ form type 1 critical curves.

Type 2. For every vertex v of the r_2-boundary, consider the

circle $\Gamma_{12}(v)$. The collection of these curves is the type 2 critical curves.

Cosider D_2 touching an edge and let X denote the position of C_2. The closest point for C_1 for a safe placement of D_1 is on type 1 curve. See figure 3.3.4. When D_2 is placed touching two adjacent edges, the closest safe placement for C_1 to avoid a collision with D_2 is on type 2 critical curve. See figure 3.3.5.

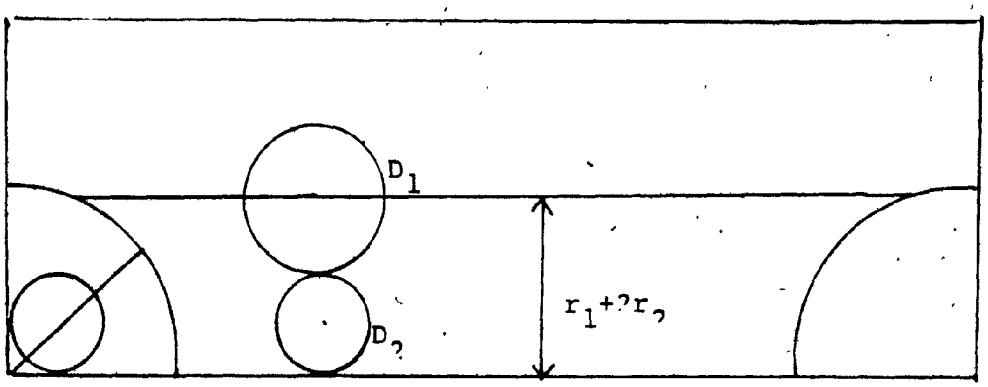


Figure 3.3.4

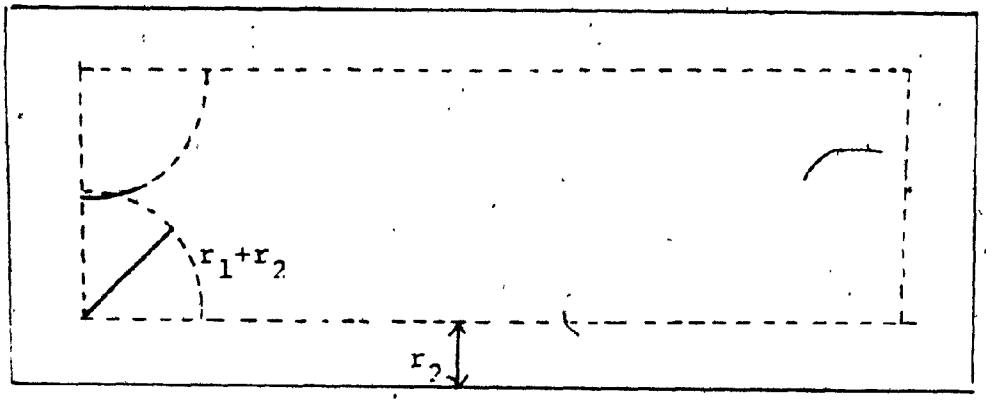


Figure 3.3.5

Based on these concepts, we describe briefly, the motion planning algorithm of Schwartz and Sharir.

Algorithm TCSS

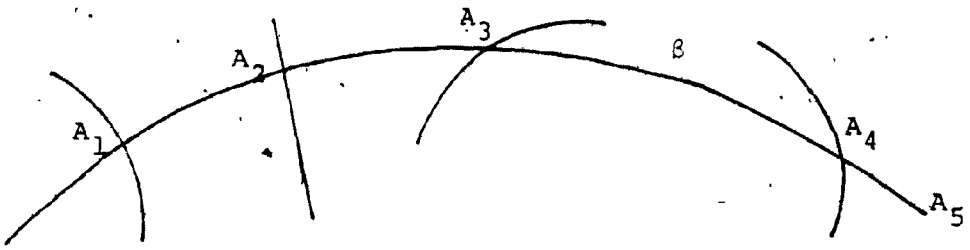
1. The edges of the boundaries W_0, W_1, ... W_{b-1} are

enlarged by the amounts r_2, r_1 and $r_1 + 2r_2$. This forms r_2-boundary, r_1-boundary and type 1 critical curves respectively.

2. All intersections of edges in r_2-boundary are computed and the circular type 2 critical curves at each corner is generated.

3. All pairs of intersections of curves in the collection C of r_1-boundary, r_2-boundary, type 1 critical curves and type 2 critical curves are computed. That is, the intersections of all grown-up boundaries are computed systematically.

4. For each curve β in C, we sort the points of intersection on β (the sorting is done fixing one direction in β). This gives a set of open segments on the curve β . See the figure below.



(A_1,A_2), (A_2,A_3), (A_3,A_4), (A_4,A_5) and (A_5,A_5) are open segments on β.

5. Let Σ denote the collection of all open segments computed in step 4. For each β' in Σ , a direction is associated and $r(\beta')$ and $l(\beta')$ designate the non-critical regions lying immediately to the right and to the left of β' respectively. Then directed segment β', $r(\beta')$, $l(\beta')$

are to be stored. In addition, for each points of intersection A_i computed on β, segments passing through A_i are sorted in counter-clockwise order in which they enter or exit A_i.

5. All non-critical regions are computed. In order to do this, a segment β' in Σ is considered and it is followed in one direction. Since β' is already directed, it can either be followed in that direction (forward) or in the opposite (backward) direction. For each β' in Σ, region(β') denotes $r(\beta')$ if the direction is forward or $l(\beta')$ if the direction is backward. Suppose, A_1 is the end-point of β' in the chosen direction. Consider the collection of segments of Σ incident at A_1. Choose β'' from this collection, which is next to β' in the sorted list. Then continue to follow β'' in the direction away from A_1. We repeat until we encounter β' again and at this stage we have a non-critical region. This procedure is repeated until all directed segments in Σ have been traversed.

7. A connectivity graph is defined in this step and is done in several stages.

7.1 For an arbitrary point X in a non-critical region R, the set

$P(X) = \{$ Y in R / Y is a free position for D_2 when D_1 is kept at X $\}$.

This is simply the set of all points on or outside $\Gamma_{12}(X)$ common to the region R.

$\sigma(X)$ = { boundaries of connected components of $P(X)$ }.

$\sigma(X)$ is invariant for all X in R and hence it can be denoted as $\sigma(R)$. Moreover, this can be computed similar to steps 1 to 5. $\sigma(R)$ is generated for all non-critical regions that are computed.

7.2 Define G, the connectivity graph as G = (V,E) where the vertex V of the graph is defined as follows :

V { (R,S) / R is a non-critical region, S in in $\sigma(R)$ }.

The edge set is defined using the following criteria :

Two vertices (R_1,S_1) and (R_2,S_2) are joined by an edge if and only if one of the following conditions hold :

R_1 is adjacent to R_2 and $S_1 = S_2 = S$ is in $\sigma(R_1) \cap \sigma(R_2)$. Call this type 1 edge.

R_1 and R_2 are adjacent and S_1 is in $\sigma(R_1) - \sigma(R_2)$ and S_2 is in $\sigma(R_2) - \sigma(R_1)$. Call this type 2 edge.

8. Given (I_1,I_2) and (F_1,F_2), the initial and final configurations, the non-critical regions R and R' containing I_1 and F_1 are found. Next, the connected components S (S') in $\sigma(R)$ ($\sigma(R')$) containing I_2 (F_2) are found. (R,S) and (R',S') are vertices in the connectivity graph.

9. If there is a path between (R,S) and (R',S') in G then there is a continuous motion between (I_1,I_2) and (F_1,F_2). Otherwise, there is no motion. The motion, if it exists, is a sequence of iterative motions determined by the edges in the path.

Informally, we explain why the existence of a path between (I_1,I_2) and (F_1,F_2) in G assures a collision free

motion for the disks from (I_1,I_2) to (F_1,F_2) in the given region. Suppose there is a path in G from (I_1,I_2) to (F_1,F_2). Consider an edge in this path. If this is a type 1 edge $<(R_1,S), (R_2,S)>$, where S is in $\sigma(R_1) \cap \sigma(R_2)$, then disk D_2 of radius r_2 can be fixed at a position inside the region S while disk D_1 of radius r_1 can be moved freely (without collision) from any position in R_1 to any position in R_2. This is a direct consequence of the definitions. Further this is an iterative motion. On the other hand if the edge under consideration is a type 2 edge $<(R_1,S_1), (R_2,S_2)>$, where S_1 is in $\sigma(R_1) - \sigma(R_2)$ and S_2 is in $\sigma(R_2) - \sigma(R_1)$, the disk D_1 of radius r_1 placed anywhere within R_1 and the disk D_2 of radius r_2 placed anywhere within S_1 can be freely moved simultaneously to arbitrary positions within R_2 and S_2 respectively. This is a simultaneous motion. Arguing like this for every edge in the path and observing that edge definition insists on the adjacency of non-critical regions R_i, we conclude that there is a collision free motion for the two disks from (I_1,I_2) to (F_1,F_2). Conversely, when there is a collision free motion for the disks from (I_1,I_2) to (F_1,F_2), we can show that there is a path in G between (I_1,I_2) and (F_1,F_2). Towards proving this, first observe that a motion for D_1 should be a sequence of continuous motions through adjacent non-critical regions. Next, we remark that for any placement X of the center C_1 of D_1, the center C_2 of D_2 must be at least at a distance r_1+r_2 from C_1 and must be within one of the

connected components of the set of free positions available to C_2. This shows as to why we need to consider edges between the vertices (R_1,S_1) and (R_2,S_2) where R_1 and R_2 are adjacent and S_1 and S_2 are as defined in step 7 of the algorithm. Once a mapping between a local motion and edge is established, global motion between (I_1,I_2) and (F_1,F_2) in the given region corresponds to a sequence of edges and the continuity of motion forces this sequence to be a path in G.

Finally, we remark on the generality, complexity and disadvantages of this algorithm. The algorithm is a particular instance of the techniques put forward by the same authors in [20] on computing the topological properties of algebraic manifolds. Here, the manifold of free configurations of the system is four dimensional. However, a decomposition strategy separates the concerns with the original four dimensional problem into two 2-dimensional subproblems. These subproblems once again belong to a continuous space but a careful study of the geometric properties of the critical curves reduces it to a combinatorial issue based on a graph model. Thus the algorithm is general, applicable to obstacles of any shape in any dimension but requires different geometric properties to be examined.

The construction of the graph, although makes a discrete solution feasible, the overall complexity still remains high. A detailed complexity analysis is not given in [19] and our investigation reveals that the overall

complexity is $O(n^5)$ and not $O(n^3)$ as claimed. A step by step cost analysis folllows :

Clearly r_2-boundary, r_1-boundary, type 1 and type 2 critical curves can all be constructed in $O(n)$ time. We assume, for example, the primitive geometric operations such as translation of a line segment can be done in constant time. Thus cost of step 1 is $O(n)$.

The intersection of adjacent pairs of enlarged edges can once again be computed in a constant time. Thus the vertices of all enlarged boundaries in step 2 can be computed in time $O(n)$.

Next in step 3, all pairs of intersections of curves in the entire collection C are computed. Since there are $O(n)$ edges in C and all mutual pairs must be examined for intersection, the cost of this step is $O(n^2)$.

The next three steps are most crucial and requires good data structures to lower the time complexity. In step 4, there is a cost for sorting the points of intersections on a curve belonging to C. Assume that the curves (either line segments or circular arcs) can be stored so as to permit accessing of

a. all curves through a given point of intersection and

b. all points of intersections lying on a curve

in an efficient manner. Then the points of intersections can be found in a sorted order while traversing the edge in a prescribed direction and finding its point of intersections with other curves. In other words, sorting is

inherent in the way we accumulate the points. Thus we do not have a separate cost for step 4.

Next we consider steps 5 and 6 together. These are two most important steps in the algorithm, for it enables the definition of G and hence its size. Through each point, a number of segments pass. Because there are $O(n^2)$ points of intersections and through each point of intersection, a maximum of n segments can pass, the number of segments is $O(n^3)$. All these segments are examined to determine all non-critical regions. Since, each segment is associated with two directions (forward and backward) and must be traversed both ways in complete determination of the regions having this segment as a common boundary and each region can have utmost $O(n)$ boundaries, the cost of determining all non-critical regions is $O(n^4)$. However, note that there are only $O(n^2)$ non-critical regions. Moreover, associated with each non-critical region R, we must find the set of all connected components $\sigma(R)$. For a region R, one connected component of $\sigma(R)$ can be determined in $O(n)$ time. Since there can be utmost $O(n)$ connected components, the cost of determining all the connected components of $\sigma(R)$ is $O(n^2)$. However, there are $O(n^2)$ non-critical regions and hence all the connected components of all the non-critical regions can be found in $O(n^4)$.

Next we examine the size of G, the connectivity graph. Clearly there are $O(n^3)$ vertices, since there are $O(n^2)$ non-critical regions and each region can have $O(n)$ connected

components in $\sigma(R)$. In order to estimate the number of edges, we must determine the number of adjacent pairs of non-critical regions and the number of edges between (R_1,S_1) and (R_2,S_2) when R_1 and R_2 are adjacent.

For a pair of adjacent non-critical regions R_1 and R_2, the edge set of type 1 is $A \cap B$ and the edge set of type 2 is (A-B) and (B-A) where $A = \sigma(R_1)$, $B = \sigma(R_2)$. One can prove that $|A-B|$ and $|B-A|$ is utmost 1 and $|A \cap B|$ is utmost $O(n)$. Hence, given one pair (R_1,R_2) of adjacent non-critical regions, the total number of edges is $|A \cap B|^2 + |A-B|\ |B-A| = O(n^2)$.

Now consider one non-critical region R_1. Since R_1 is closed, its boundary can have utmost $O(n)$ edges. Every edge can be common to two regions of which one is R_1. Hence R_1 can be adjacent to $O(n)$ non-critical regions. Since there are $O(n^2)$ non-critical regions, there are $O(n^3)$ mutually adjacent pairs of non-critical regions and we have shown that each one of them can contribute to $O(n^2)$ edges in G. Thus the number of edges in G is $O(n^5)$.

Finally, the path finding algorithm of Tarjan [23] can find a path in G between (I_1,I_2) and (F_1,F_2) in time $O(|V|+|E|) = O(N^5)$.

We also remark that a straightforward generalization of this algorithm to three disks has complexity $O(n^{13})$ and generalization to $k \geq 4$ disks is not possible.

Yap [14] gives a method based on the principle of retraction for the coordinated motion of two and three

disks. The time complexity of this algorithm is $O(n^2)$ for two disks and $O(n^3)$ for three disks. However, this algorithm, as described in the report, considers only restricted situations for the placement of the obstacles. In addition to such a restriction, it lacks generality in the sense that for $k > 3$ disks, this algorithm is not applicable.

In the next three chapters, we give algorithms for the coordinated motion of $k \geq 1$ disks of time complexity $O(n^k)$.

Chapter 4

Motion planning for a single disk

4.1 Introduction

We first present an algorithm with time complexity $O(n^2)$ which requires simple list manipulation in a practical programming environment. When all the obstacles lie in a bounded region of constant width, we propose an algorithm with time complexity of $O(n)$.

4.2 An $O(n^2)$ algorithm for the motion of a single disk

Let W_0 denote the boundary of the outer wall. The polygonal shape of this boundary is represented as a set of K vertices V_1, V_2 .. V_k in the Euclidean plane. We assume that the vertices are given in counter-clockwise direction so that when the boundary is traversed, the set of free positions for the disk is to the left of the boundary. For each inner boundary, the vertices are taken in clockwise direction so that the set of free positions is to the left of the boundary. Further, the boundaries are assumed to be closed. Let r be the radius of the disk, I be the given initial position of the disk, F, the final position to which the disk is to be moved and let there be J obstacles whose boundaries are to be avoided in a motion.

During the motion of the disk D amidst the barriers, D can touch but should not collide with any of the boundary. Hence, the minimum distance between the center C of the disk and an arbitrary point on any of the boundary must at least be r. Hence, we find the set of free positions FP for C in

the configuration where the boundaries are enlarged by r. The rules for creating the grown-up configuration are :

1. Straight edges are displaced by r so that an edge AB would be replaced by a parallel edge A'B' at a distance r. The displacements are effected towards the free positions of the disk.

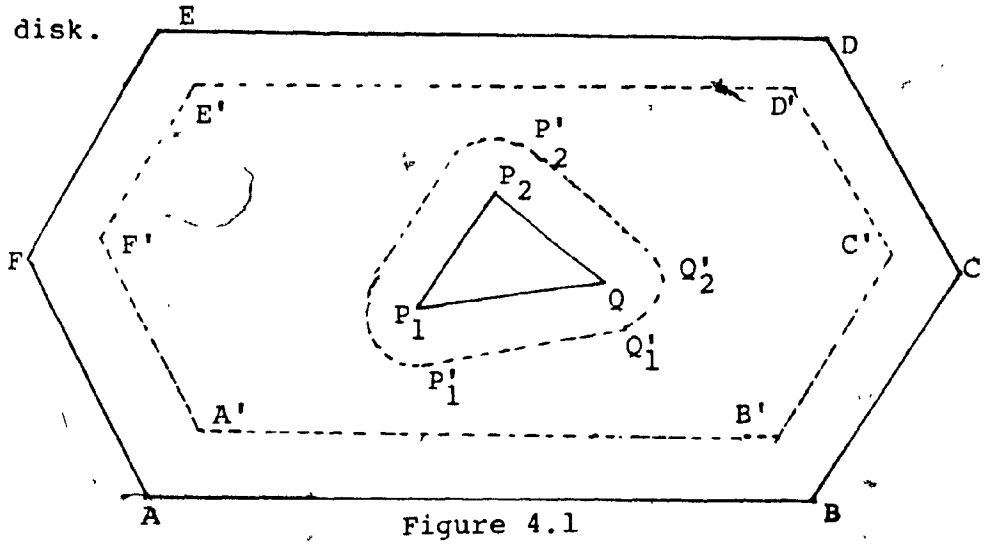


Figure 4.1

2. Convex corners are replaced by circular arcs of radius r.

The set of vertices for any enlarged boundary is determined by the above two rules. To be more specific, if an original vertex is not convex, it is replaced by the point of intersection of the corresponding displaced edges. For each convex vertex of a boundary, a circular arc and two new vertices are introduced in the enlarged boundary. It is clear that the positions for the center of the disk is the union of the closed connected components of the region bounded by the displaced edges and corners. For example, suppose Q is a convex vertex and $E_1 = (P_1, Q)$ and $E_2 = (P_2, Q)$ are the two edges incident at Q. Let $E_1' = (P_1', Q_1')$

and $E_2' = (P_2', Q_2')$ are the displaced edges of E_1 and E_2. Then Q_1' and Q_2' are the new vertices. The circular arc center at Q, radius r and bounded by the radius vectors QQ_1' and QQ_2', taken in the clockwise direction from Q_2' to Q_1' is the new edge. The other edges and vertices in figure 4.1 are obtained in a similar fashion.

Next, we shall discuss a method to partition the set of free positions of C into connected components.

Some of the grown-up obstacles might intersect the enlarged outer boundary. When this happens they have to be identified and unified to get a new outer boundary. This can be accomplished by checking whether any of the edges of the inner boundaries intersect the outer boundary. Points of intersection, if any, can be introduced as new vertices in the corresponding edges of both boundaries. Once this computation is done for the edges of the outer and an inner boundary, the unification method to be described below is effected :

A vertex of outer boundary which does not lie inside the inner obstacle is chosen. From that vertex, the outer boundary is traversed in the counter clockwise direction until a new vertex, say P_1, is encountered. This sequence of vertices will become part of the new outer boundary. Since the new vertex is a point of intersection, this has already been introduced in the corresponding edge of the inner obstacle. Consider the vertex Q next to P_1 on the inner boundary. If Q does not belong to FP of C then the

outer boundary is traversed from P_1 in the counter-clockwise sense and the above method is repeated. However, if Q belongs to FP of C, we traverse the inner boundary in the clockwise direction until the next new vertex is encountered. This sequence of new vertices and edges is appended to the partially formed outer boundary. This method is repeated until the original vertex with which we started, is reached. On termination, we have a connected component bounded by the outer boundary. It is possible that the region bounded by the outer boundary gets divided into several components by an inner boundary. Hence, in order to capture all connected components, the above method must be repeated until all the vertices of the outer boundary that belong to FP occur among the components.

The unification method must be repeated for every inner boundary intersecting the new outer boundary. On completion of this process, we would have formed all connected components of the region enclosed by the outer boundary. The example in figure 4.2 illustrates this unification method :

Those inner boundaries which do not intersect the outer boundary should be tested for possible intersection among themselves. In case any two of them intersect, the same unification procedure can be used to get unified inner boundaries. Thus, the set of free positions for center C gets divided into different components where each component is a multiply connected region. Figure 4.3 shows the

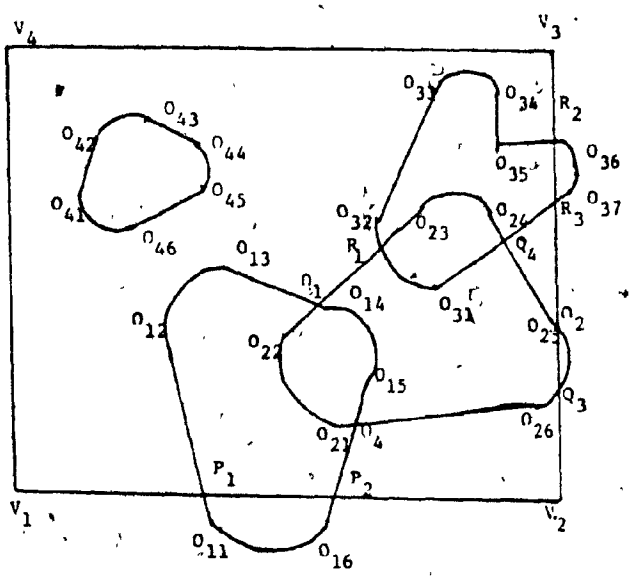


Figure 4.7

Outer with O_1: $V_1P_1P_2V_2V_3V_4V_1$; Inner with O_1: $O_{11}P_1O_{12}O_{13}O_{14}O_{15}P_2O_{16}O_{11}$

New Outer: $V_1P_1O_{12}O_{13}O_{14}O_{15}P_2V_2V_3V_4V_1$

New Outer with O_2: $V_1P_1O_{12}O_{13}Q_1O_{14}O_{15}Q_4P_2V_2Q_3Q_2V_3V_4V_1$

Inner O_2 : $Q_{21}O_{22}Q_1O_{23}O_{24}O_{25}Q_2Q_3O_{26}Q_4O_{21}$

New-outer 1: $V_1P_1O_{12}O_{13}Q_1O_{23}O_{24}O_{25}Q_2V_3V_4V_1$

New-outer 2: $Q_4P_2V_2Q_3O_{26}Q_4$; New-outer 3: $R_4Q_2R_3R_4$

New-outer 1 with O_3: $V_1P_1O_{12}O_{13}O_1R_1O_{23}O_{24}R_4O_{25}Q_2R_3R_2V_3V_4V_1$

New-outer 1: $V_1P_1O_{12}O_{13}Q_1R_1O_{31}O_{32}O_{33}O_{34}R_2V_3V_4V_1$

regions resulting from the unification method applied to the enlarged obstacles of figure 4.2.

Having identified these regions, the next step in this method is to test if the initial position I and the final position F of the disk belong to the same component. If

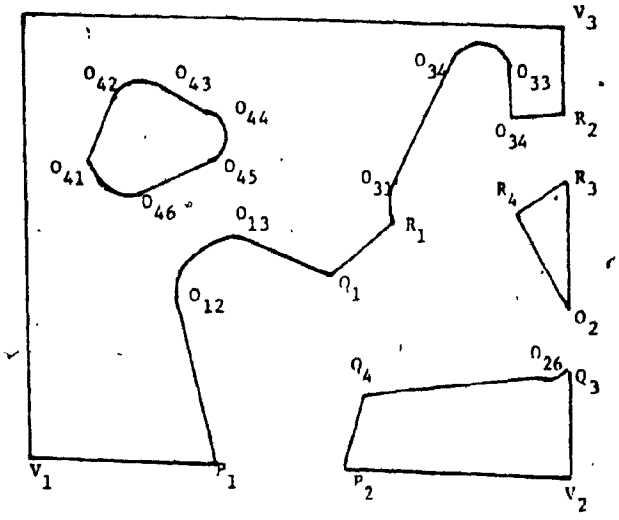


Figure 4.3

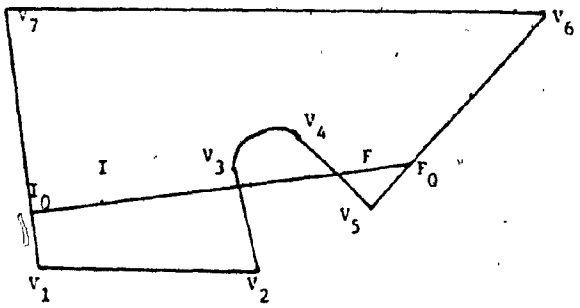


Figure 4.4

they belong to two different components then there is no motion for the disk from I to F. When I and F belong to the same region, motion from I to F can be determined very easily. Let I_0 and F_0 be the external points of intersection of the line IF with the boundary. Let us denote the motion of C from X to Y by (X, Y). Note that this motion is a translation if (X, Y) is a straight edge; otherwise, the motion is a rotation. ((I, I_0), I_0 to F_0 along the boundary, (F_0, F)) gives a motion for the disk to

reach its destination. See figure 4.4.

The two major steps in method 1 are enlarging the boundaries of obstacles and unification of enlarged boundaries to determine the connected components. Each major step involves a finite number of primitive operations such as : 1. displacing each straight edge by a constant amount parallel to its original direction; 2. determining circular edges and 3. determining whether or not, a vertex of a region is convex. Since each primitive operation involves a constant number of additions, multiplications or comparisons, the cost of the method will be measured by the number of primitive operations required to determine the existence of a motion.

The cost of the first step namely that of growing the obstacles and the boundary is O(n) where n is the total number of vertices in the obstacles and the outer boundary. The number of vertices in the grown-up configuration is also O(n) since we can have utmost 2n vertices and 2n edges in the grown-up boundaries. Next, in the unification step, the grown-up obstacles are checked for possible intersections. Each edge, in the worst case , would be tested with every other edge for intersection. Once an intersection is detected, corresponding boundaries are unified together by testing each new vertex with respect to the set of free positions and traversing each edge of a boundary at most once. Hence, the unification process is linear on the

number of vertices of the unified boundaries. So, the cost of detecting intersections and unification for all boundaries is $O(n^2)$. The cost of determining the component(s) to which I and F belong is linear in the total number of edges. Since there are at most $O(n^2)$ edges in the new configuration, this cost is $O(n^2)$. Finally, if they belong to the same region, I_0 and F_0 can be determined in time O(n) and so the path is determined in O(n) time. Hence, we conclude that the total cost of this method is $O(n^2)$.

4.3 Representation of planar regions and contours :

In this section, we shall describe the region representation technique developed by Merrill [13]. This representation facilitates efficient methods for computing boundary intersections, unions as well as for point enclosure problems. This method when adopted to motion finding problem, reduces the overall complexity of the method.

In [13], a rectangular grid is used within which regions are to be represented. In order for a region boundary to be represented in this grid, it should satisfy three conditions : 1. Boundary should be closed and continuous. 2. If a grid line parallel to the reference axis passes through extreme points (local maximum or minimum points) or points of inflection, the number of grid points of the boundary on this line may violate Jordon Curve Theorem. In case, it happens, these special points have to

be repeated certain number of times. 3. The boundary should not loop back on itself.

Once the boundary is made to satisfy these conditions, it can be represented as a discrete data in the grid. Let ymin(T) and ymax(T) denote the minimum and the maximum Y co-ordinates of a boundary T to be represented in the grid and let p = ymax(T) - ymin(T) + 1. Then for each y_i where y_i = ymin(T) + i - 1, $1 \leq i \leq p$, let $Y_i(T)$ denote the set of all X-coordinates of the points of the contour T whose y-coordinate is y_i. This set, called Y-partition corresponding to y_i, is kept in a sorted fashion. In this manner, the entire boundary T is discretized into sets of Y-partitions. This representation enables simplified solutions to two primitives that we need.

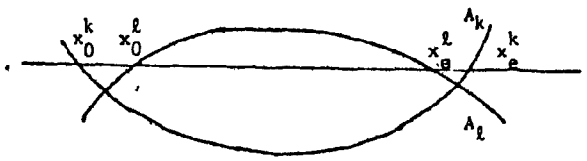
Primitive 1 : Given two regions A_k and A_l, compute the region $A_k - A_l = A_m$ and its components when A_l is not contained in A_k. Further, determine the enclosing component of I anf F.

A solution to this primitive is not given in [13]. Our solution is as follows : Consider ymin(m) = max { ymin(k), ymin(l) }, ymax(m) = min { ymax(k), ymax(l) }. Regions A_k and A_l could overlap only in the strip determined by ymin(m) to ymax(m). Those Y-partitions of region A_k corresponding to ymin(m) to ymax(m) should be modified to get corresponding Y-partitions of region $A_k - A_l$. Other Y-partitions of region A_k remain same for region $A_k - A_l$.

In the common range that is computed, each odd-even pair of

every Y-partition in region A_k should be analyzed in the following manner and points for region $A_k - A_l$ should be found. Suppose $(x_0(k), x_e(k))$ is an odd-even pair of region A_k and $(x_0(l), x_e(l))$ is an odd-even pair of the region A_l in an Y-partition $Y_s(m)$ where $ymin(m) \leq y_s \leq ymax(m)$, then only one of the following 5 situations would arise :

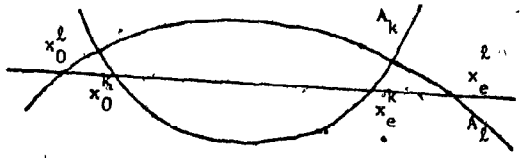
1.



$(x_0(k) \leq x_0(l))$ and $(x_e(l) \leq x_e(k))$

In this case, $(x_0(k), x_0(l))$ and $(x_e(l), x_e(k))$ are introduced as two consecutive odd-even pairs in $A_k - A_l$.

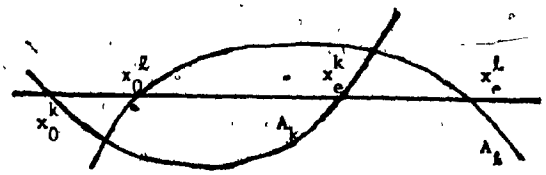
2.



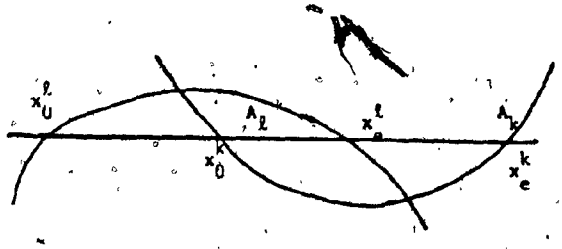
$(x_0(k) > x_0(l))$ and $(x_e(k) < x_e(l))$

No pair is introduced in region $A_k - A_l$ corresponding to $(x_0(k), x_e(k))$ and $(x_0(l), x_e(l))$.

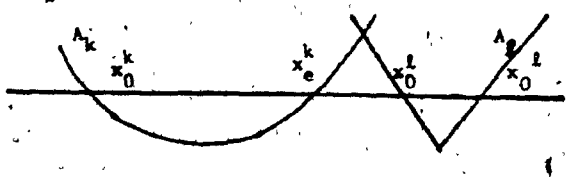
3. $(x_0(k) \leq x_0(l))$ and $(x_e(k) < x_e(l))$.



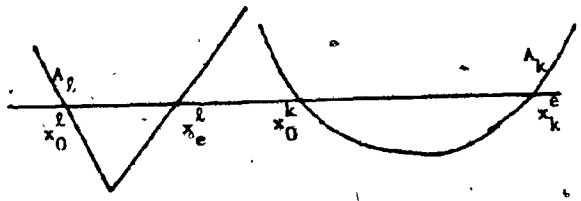
$(x_0(k), x_0(l))$ is the only odd-even pair introduced in the corresponding partition of $A_k - A_l$.



4. $(x_0(k) > x_0(l))$ and $(x_e(l) \leq x_e(k))$.
In this case, $(x_e(l), x_e(k))$ is introduced as an odd-even pair in $A_k - A_l$.



5. $(x_0(l) \geq x_e(k))$ and hence $(x_0(k), x_e(k))$ is an odd-even pair in $A_k - A_l$.
5. $(x_e(l) \leq x_0(k))$ and so $(x_0(k), x_e(k))$ is an odd-even pair in $A_k - A_l$.



By analyzing the partitions of the two regions A_k and A_1 in the common range, as explained, region boundary for $A_k - A_1$ is formed. During the above analysis, if either (O_1, E_1) or (O_2, E_2) (odd-even pairs containing the initial and final positions) is changed then the new odd-even pair is computed and kept in the corresponding pair.

In the solution to our problem, we need Y-partitions of region $A_k - A_1$ only when A_1 is not contained in A_k. Region A_1 is not contained in A_k when one of the following conditions hold :

1. ymin(1) < ymin(k)
2. ymax(1) > ymax(k)
3. One of the situations 2, 3, 4, 5, 6 arises when analyzing the Y-partitions in the common range.

In case, when one of the above conditions holds, different components of the region $A_k - A_1$ can be determined by the following method :

Consecutive Y-partitions of the region $A_k - A_1$ are analyzed in the following manner to obtain the different components. Suppose $Y_i(m)$ and $Y_{i+1}(m)$ are two consecutive Y-partitions of the region $A_k - A_1$. If there is an odd-even pair

$(x_0(i+1), x_e(i+1))$ in $Y_{i+1}(m)$ such that interval $(x_0(i+1), x_e(i+1))$ is completely contained in an even-odd pair $(x_e(i), x_0(i))$ of $Y_i(m)$ then the pair $(x_0(i+1), x_e(i+1))$ may indicate the beginning of a new component. Corresponding odd-even pairs of successive partitions are analyzed. After certain Y-partitions, if there is a $Y_j(m)$ such that there is an odd-even pair $(x_0(j), x_e(j))$ in $Y_j(m)$ and it is completely contained in an even-odd pair $(x_e(j+1), x_0(j+1))$ of $Y_{j+1}(m)$, then a component of region $A_k - A_l$ is recognized. The y-values of this component range from y_{i+1} to y_j and the Y-partitions of this component are as follows :

$Y_{i+1}(m) = \{x_0(i+1), x_e(i+1)\}$; For y_w such that $y_{i+2} \leq y_w \leq y_{j-1}$, $Y_w(m) = \{$ corresponding odd-even pairs from Y(w)-partition of the region $A_k - A_l \}$ and $Y_j(m) = \{x_0(j), x_e(j) \}$

If these Y-partitions contain both (O_1, E_1) and (O_2, E_2) then this is the enclosing component of I and F. If this set has only one of the pairs then I and F belong to two different component and hence there is no motion. If both of them do not belong to this component then this component does not contain I and F. These odd-even pairs are removed from the corresponding Y-partitions of the region $A_k - A_l$. By, analyzing all the Y-partitions of the region $A_k - A_l$, Y-partitions of the component containing I and F can be determined.

Primitive 2 : Given two regions A_k and A_l, compute their

union A_m if they intersect :

The boundary of the union of the regions A_k and A_l will have Y-partitions from ymin(m) = min { ymin(k), ymin(l) } to ymax(m) = max { ymax(k), ymax(l) }. Rules for creating Y-partitions of the common region are formulated in a manner analogous to that for creating $A_k - A_l$ and they are summarized as follows :

1. If ($x_e(k) \leq x_0(l)$) then ($x_0(k)$, $x_e(k)$) and ($x_0(l)$, $x_e(l)$) are considered two adjacent odd-even pairs in the union.

2. If ($x_0(k) \geq x_e(l)$) then ($x_0(l)$, $x_e(l)$), ($x_0(k)$, $x_e(k)$) are two adjacent odd-even pairs.

3. If ($x_0(k) < x_0(l)$) and ($x_e(k) \leq x_e(l)$) then ($x_0(k)$, $x_e(l)$) is introduced as a pair.

4. If ($x_0(k) > x_0(l)$) and ($x_e(k) > x_e(l)$) then ($x_0(l)$, $x_e(k)$) is introduced as an odd-even pair.

5. If ($x_0(k) \leq x_0(l)$) and ($x_e(l) \leq x_e(k)$) then ($x_0(k)$, $x_e(k)$) is a new pair in $A_k - A_l$.

6. If ($x_0(l) \leq x_0(k)$) and ($x_e(k) \leq x_e(l)$) then ($x_0(l)$, $x_e(l)$) is a new pair.

The two regions intersect if and only if the following conditions are true :

1. A_k and A_l have at least one common y-line. 2. While analyzing the Y-partitions of the common y-lines, at least one of the situations 3, 4, 5 or 6 arises in one of them.

In the following section, we make use of this discrete representation and primitives for an efficient solution to

the motion finding problem.

4.4 A fast algorithm for planning the motion of a disk.

Outer boundary and boundaries of inner obstacles are grown-up by the amount R and the enlarged regions are represented as Y-partitions. Let (xmin, xmax, ymin, ymax) denote minimum and maximum X and Y co-ordinates of the original outer boundary. Then the rectangle $\{(xmin - R, ymin - R), (xmax + R, ymax + R)\}$ encloses all the boundaries. In this rectangle, y-lines are introduced at unit distance from ymin - R to ymax + R and also through I and F. Further, for each boundary T, let $P_1(T) = \{xmin(T), y_1\}$, $P_2(T) = \{xmax(T), y_2\}$, $P_3(T) = \{x_1, ymin(T)\}$ and $P_4(T) = \{x_2, ymax(T)\}$. Then y-lines are also introduced through these points. This would capture the intersection of any two regions when the common region is within two consecutive y-lines.

Regions to be represented in the rectangle should satisfy the three conditions mentioned in section 4. Since all our boundaries are closed, condition 1 is always satisfied. Boundaries of the obstacles in our problem never loop back on itself and so situation of this type would never occur. As the Y-partitions for each region are determined extreme points and points of inflection are detected and condition 2 is checked among those points.

It is sufficient to describe how the Y-partitions of each boundary is formed. Each boundary T is traversed once

to get (ymin(T), ymax(T)) of that boundary. Y-partitions of that boundary are to be determined for each y_t , ymin(T) $\leq y_t \leq$ ymax(T).

Points of intersection of any y_t -line with the edges of the boundary forms the corresponding Y-partition. If y_t is a tangent to a circular arc then the point of intersection is introduced twice. If y_t passes through a vertex V such that edges incident on V are both above or below y_t (local minimum or maximum) then V is repeated twice. When one of those edges is above and another is below y_t then V is a point of inflection and it is added only once. Another situation in which points of inflection would arise is when an edge PQ ($P_x \leq Q_x$) of the region coincides with y_t. If T is an inner boundary then we claim that the interval of y_t that precedes P (if exists) and the interval that follows Q (if exists) both belong to the region T. The proof is omitted.

As a consequence of our claim for an inner boundary T, if an edge of T say PQ coincides with y_t then the following method is adopted to form $Y_t(T)$.

All points of intersection of the obstacle with y_t is determined and sorted in ascending order. If the sorted sequence is of the form ... *** P' P Q Q' **** .... then intervals [P',P], [P, Q] and [Q, Q'] belong to the region T and so all three pairs should be odd-even.

We consider the case in which region T is the outer boundary and an edge PQ coincides with y_t. Now, we claim

that the interval that precedes P (if exists) and the interval that follows Q (if exists) both do not belong to the region. Once again we omit the proof.

Hence, we conclude that if contour T is the outer boundary and if one of its edges coincides with y_t then all points of intersections of T with y_t is determined. If

...*** P' P Q Q' *** ... is the sorted sequence of $Y_t(T)$ then (P', P) and (Q, Q') should be even-odd pairs (do not belong to the region) whereas (P, Q) should be an odd-even pair. In this fashion, Y-partitions for all grown-up boundaries are formed. From the Y-partitions of the outer boundary, the two odd-even pairs (O_1, E_1) and (O_2, E_2) containing I and F can be determined.

Having found the Y-partitions for the outer boundary and an inner boundary O, the next step is to check whether the inner boundary penetrates the outer boundary. If the inner boundary O is completely contained in the outer boundary then it is kept in a set S. S contains those inner boundaries which do not intersect the outer boundary and this set is initially empty. If O intersects any of the member of this set, then union of those boundaries are formed. This task is done by primitive 2.

If the inner boundary penetrates the outer boundary, they have to be merged together to get a new outer boundary. In this new outer boundary, if I and F belong to two different components then there is no motion. Otherwise, enclosing component of I and F should be determined. All

the above-mentioned tasks can be done using primitive 1. If I and F belong to the same component, the new outer boundary is this component and the above process should be repeated for every inner boundary.

At the end of the process, if I and F belong to the same component then each member of the set S should be checked with this component for possible penetration. Those penetrating members should be merged with the component.

On completion of this method, either we would be having the component containing I and F, proving the existence of motion or the method would have stopped due to I and F belonging to two different components. Thus the decision problem is solved using the partitioned representation and the primitives.

Analysis of Method 2.

Finally, we shall analyze this algorithm to find its cost.

Cost of the first step namely growing-up the boundary is O(n). In the next step, the Y-partitions of each boundary are determined. We have a maximum of (ymax - ymin + 3) + 2R + 4J lines and each line can intersect utmost O(n) edges of the boundaries. So, the cost of forming the Y-partitions of all the boundaries is O(n). These points are to be kept in sorted order in each y-line. Sorting can be done as follows :

When the Y-partitions of an inner boundary are being formed, each y-line is tested for possible intersection with every

edge of that boundary. We can have utmost n points of intersection on a y-line with any boundary. As a point (x, y) is found out, it is put into the location (x - xmin (T)) of an array containing xmax(T) - xmin(T) + 1 locations, where xmin(T) and xmax(T) are the minimum and maximum of the X-coordinate of that boundary T. Due to the special nature of our problem, this sorting is linear on the number of points.

Having formed the Y-partitions, inner boundaries which penetrate outer boundary are merged with the outer boundary and this is accomplished using primitives 1 and 2. In the worst case, (situation when the inner boundaries are pairwise disjoint and none of them penetrates the outer boundary) primitives 1 and 2 would be used J(J+1)/2 times. Let us first compute the cost of doing primitive 1 once. Let the outer boundary W and an inner boundary O have m_1 and m_2 vertices respectively. On a common y-line, the Y-partitions of W and O can have a maximum of m_1 and m_2 points. The cost of forming one Y-partition for region W-O in the common range is $m_1/2 + m_2/2 - 1$ which is less than $2n/2$. Hence, the total cost of forming region W-O is a constant multiple of n which is O(n). From the Y-partitions of region W-O, extracting the Y-partitions of the component containing I and F is also O(n). Hence, the cost of doing primitive 1 is O(n). In the same way, we can prove that the cost of doing primitive 2 is O(n). Primitives 1 and 2 would be used utmost J(J+1)/2 times. Hence the cost of

determining enclosing component of I and F and thereby solving the decision problem is utmost O(n).

4.5 Remarks

Among all the motion planning problems, probably we have addressed the simplest case. Yet, as we have demonstrated, an efficient solution requires new techniques in discrete topology and computational geometry. In [14], an O(n logn) motion planning algorithm based on the construction of Voronoi diagram is given. Our main thrust in this paper has been to speed up the solution by considering discrete versions of continuous spaces. The representations for contours and the regions bounded by them provide rapid computational procedures for testing algebraic and topological properties of the region. This representation may not be valid in general situations; however, within the context of determining components of regions bounded by the enlarged boundaries of obstacles, the inherent approximation of representation does not affect the accuracy. In other words, connected components are always determined correctly and hence the decision problem is solved exactly. Moreover, an optimal algorithm for solving the point enclosure problem requires O(n) steps when the region is a polygon with n vertices. Hence, our algorithm optimally solves the motion planning problem for a single disk. The only question left open by us is that of finding a path from I to F. We assert that this can be done in time O(n) if the edge information for each point in the

Y-partitions is kept.

There are several advantages to our algorithm. Some of these are : 1. Having determined the existence of motion for a given set of obstacles, if one or more new obstacle is introduced, our method can be extended with minimum cost to determine the existence of motion in the new configuration. 2. If the initial position I and the final position F are not known in advance, then all the connected components are needed to determine the motion between arbitrary points of FP. Our method can be modified to handle this case without changing the overall complexity.

When an obstacle is deleted from the set of obstacles, the resulting set of free positions and hence the set of connected components will change. A similar situation would arise when a disk of different radius is considered while fixing the boundaries, the initial and the final positions. In all these cases, we do not yet know how our algorithm can be minimally modified. These restrictions apply to all known algorithms for this problem; however, the first advantage mentioned above applies only to our algorithm.

Chapter 5

Coordinated motion of two disks

5.1 Introduction

Schwartz and Sharir [21] give a general algorithm to solve the coordinated motion of two and three disks moving among polygonal barriers, with respective time complexities $O(n^5)$ and $O(n^{13})$. Recently, Yap [25] has given $O(n^2)$ and $O(n^3)$ algorithms to solve the coordinated motion of two disks and three disks moving among polygonal barriers with some restrictions. His method is based on the principle of retraction and cannot be generalized for $k > 3$ disks. The algorithm to be discussed in this chapter has time complexity $O(n^2)$ for the motion of two disks.

The closest contender to our algorithm is due to Schwartz and Sharir [21]. The general philosophy of our algorithm is similar to [21]; yet there are substantial differences in the way that the set of free positions for the disks are identified, handled and partitioned into components. Our method is more like 'divide and conquer' approach, separating the concerns and dealing with the components as independently as possible. When the components are to be found, we also take a 'directed goal' approach. Once we reach the stage wherein each (I_j, F_j) belong to a component R_j for all $j=1,2, \ldots k$, we transform the topology of the configuration to a simple combinatorial issue based on a related graph G. The existence of a motion is then related to the existence of a specific path in G.

We remark that the graph $G = (V,E)$ is constructed so that $|V| = n^k$, $|E| = n^k$ and the algorithm due to Tarjan[23] finds a path between any two specified vertices in G in time $O(n^k)$.

In our discussions we represent each wall by a set of vertices with the convention that for the outer wall W_0, the vertices are kept in counter clockwise direction and for each inner wall W_j, $1 \leq j \leq p-1$, the vertices are kept in clockwise direction. Hence traversing this representation, the free positions for the disks is to the left of the boundary traversed. At any instant, $(C_1, C_2, .. C_k)$ denote the centers of the k disks and this is a feasible configuration if $E_d(C_i,C_j) \geq r_i+r_j$ for all different pairs of disks and no disk can penetrate a wall. We also need the notion of Hausdorff distance which we define as follows : 1. For any edge e (straight edge or a circular edge) and a point P, the Hausdorff distance of e from P is $H_d(P,e) = \min_{x \in e} \{E_d(P,x)\}$

2. For two edges e_1 and e_2, the Hausdorff distance of e_1 from e_2 is $H_d(e_2,e_1) = \max_{p \in e_2} \{H_d(P,e_1)\}$

3. The distance between two edges e_1 and e_2 is $H(e_1,e_2) = \min \{ H_d(e_2,e_1), H_d(e_1,e_2) \}$

5.2 Algorithm for coordinated motion of two disks

In this section, we consider the special case of two circular disks moving amidst polygonal barriers. The important step in our approach is to determine the connected components of collision free configurations of the disks.

Once this is done the existence of a motion is determined using a graph-theoritical approach. Of course, the first basic step is due to Lozano-Perez and Wesley [12].

We transform the given configuration of polygonal obstacles so that in the transformed space, it is sufficient to determine the coordinated motion of point disks (C_1, C_2) initially placed at (I_1, I_2). Note that if the boundaries are enlarged by r_1, then the region bounded by these grown-up boundaries is the set of free positions for C_1, ignoring the presence of the other disk. In the same way, when the given boundaries are grown up by r_2, the region enclosed by these enlarged boundaries determine the set of free positions for C_2.

Next, we give a method to enlarge the given boundaries by r_1. Straight edges are displaced by the amount r_1 where the direction of displacement is towards the free positions. A convex vertex is replaced by a circular arc centred at the vertex and radius r_1. The two end points of the arc are the points of intersection of the circle with the corresponding two displaced edges. Having displaced each edge and convex vertex, the set of vertices for each enlarged boundary is obtained by the points of intersection of adjacent enlarged edges. In this fashion, the enlarged configuration G_1 for the first disk is obtained. Similarly, we determine the configuration G_2 for the second disk by enlarging the boundaries of the given obstacles by r_2.

Having determined the sets of free positions G_1 and G_2

for C_1 and C_2, next we shall compute the connected components of G_1 and G_2. In an enlarged configuration, some of the grown-up inner boundaries may intersect the grown-up outer boundary and thereby dividing the set of free positions into several components. If the initial and final positions of a disk belong to two different components of its free position then it is not possible to move the disk to the final position. Hence, in order to determine a motion it is necessary to determine the different components in each enlarged configuration. This can be accomplished as follows : Each edge of an enlarged inner boundary is checked with every edge of the outer boundary for possible intersections. Points of intersections, if any, are introduced as new points in the corresponding edges of both the boundaries.

A vertex of the outer boundary which is not inside the inner boundary is chosen and the outer boundary is traversed in the counter clockwise direction until a new vertex say P is encountered. This sequence of vertices becomes part of the new outer boundary. Then the inner boundary is searched for vertex P. If the vertex next to P in the inner boundary is not within the outer boundary then the traversal is continued in the outer boundary and the above procedure is repeated. However, if the vertex next to P is within the outer boundary then the inner boundary is traversed in clockwise direction until the next new vertex Q is encountered. The sequence of vertices from P to Q in the

inner boundary is appended at the end of the partly formed new outer boundary. Now, the outer boundary is traversed from Q and the above procedure is repeated until the original vertex with which we started is encountered. This gives one connected component. By repeating the above procedure until all vertices of the grown-up outer boundary are included in the component boundaries, we determine the connected components arising out of the intersection of one inner boundary with the outer boundary.

By considering each inner boundary with the newly formed outer boundary (components) and by repeating the above process, the set of components of the free position for a disk can be obtained. Those inner boundaries which do not intersect any of these components may have intersections among themselves. We identify these and merge together all intersecting inner boundaries to get unified inner boundaries. This can be accomplished using the same procedure described above. Any one unified boundary should be completely contained in a component determined earlier. Hence, the components of the region bounded by the outer boundary constitutes the components of the set of free position provided regions determined by completely contained inner boundaries are avoided. Thus the set of free positions is divided into different components where each component can be a multiply-connected region.

By independently applying this procedure to G_1 and to G_2, we determine the connected components of each. If at

any stage of this region finding, either I_1 and F_1 belong to two different components of G_1 or I_2 and F_2 belong to two different components of G_2 there is no motion for the disks. So consider the situation when I_1 and F_1 belong to a component R_1 of G_1 and I_2 and F_2 belong to a component R_2 of G_2. The existence of a coordinated motion can be determined by examinig the relative configurations of R_1 and R_2. We first prove that only two situations can arise and then characterize the existence of motion in each case.

Theorem 1 : Either $R_1 \subset R_2$ or $R_1 \cap R_2 = \emptyset$.

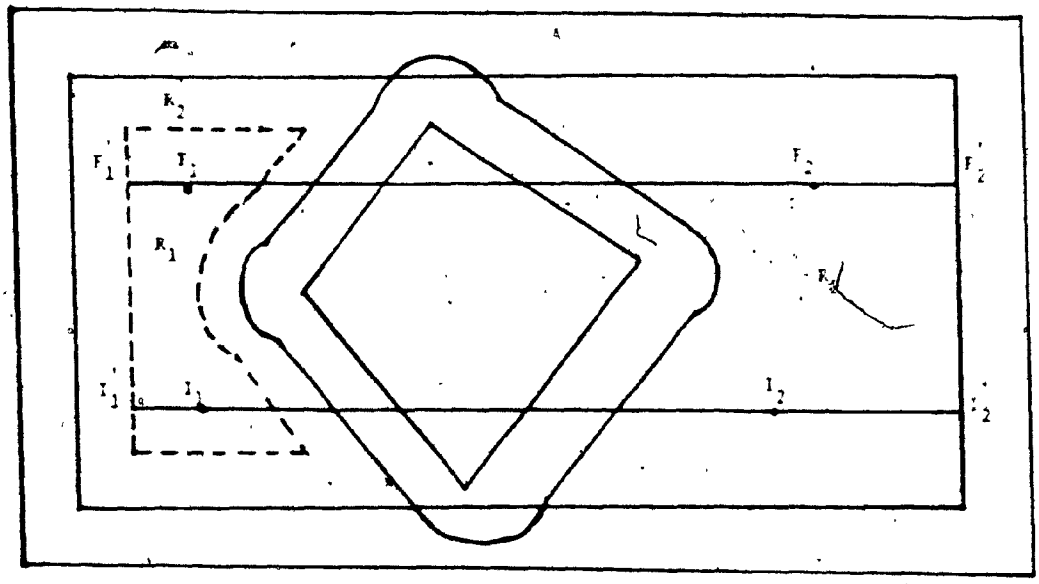
Proof : First observe that every free position for the larger disk is a free position for the smaller disk. Hence every component of G_1 must be completely contained in or disjoint from a component of G_2. Now, suppose $R_1 \cap R_2 = \emptyset$. Then the theorem is proved. Consider the case in which $R_1 \subset R_2 \neq \emptyset$. We have to prove that $R_1 \subset R_2$. This is equivalent to proving $R_1 - R_2 = \emptyset$. Let us assume the contrary that $R_1 - R_2 \neq \emptyset$. This implies that there exists a point x in $R_1 - R_2$ such that x is in R_1 and x is not in R_2. Our hypothesis is $R_1 \cap R_2 \neq \emptyset$. Hence there is a common point x' in $R_1 \cap R_2$. x and x' are in R_1 and R_1 is a connected component implies that x and x' are free positions for C_1 and they are connected by a path say P_1 in R_1. Since, $r_1 > r_2$, any free position of C_1 is a free position for C_2. Thus x and x' are free positions for C_2 and they are connected by the same path P_1. This gives that x and x' are connected in R_2 where x is not in R_2 which is a contradiction to the fact that R_2

is a connected component. This proves our claim $R_1 \subseteq R_2$.

Next, we consider the situation when $R_1 \cap R_2 = \emptyset$ and show that there is always a motion for the disks.

Theorem 2 : If $R_1 \cap R_2 = \emptyset$ then there is a motion for the disks from (I_1, I_2) to (F_1, F_2) where I_1 and F_1 are in R_1 and I_2 and F_2 are in R_2.

Proof : Because each point x in R_1 is a free position for C_2, there is a component R_2' of G_2 with $R_1 \subseteq R_2'$. We claim that $R_2' \cap R_2 = \emptyset$. For otherwise, R_2, R_2' and R_1 will be contained in $R_2' \cup R_2$ and this contradicts the fact that R_2 and R_2' are maximal. Now, we show the existence of a motion by constructing a collision free path for the disks. See Figure 5.1. Join I_1, I_2 and let it meet R_1 boundary at I_1' and R_2 boundary at I_2'. Join F_1, F_2 and let it meet R_1 boundary at F_1' and R_2 boundary at F_2'.



Note that there are parts of R_1 boundary and R_2 boundary which are at maximum distance from each other. Let these

portions be R_1' and R_2'. It is easy to see that the first disk at C_1 can be moved from I_1 to I_1' and C_2 can be moved from I_2 to I_2' simultaneously. Now, C_1 can be moved along R_1' upto F_1' and C_2 can be moved along R_2' upto F_2' and this may be done simultaneously. Finally, C_1 can be moved from F_1' to F_1 and C_2 can be moved from F_2' to F_2. Hence, the proof is complete.

Next, we consider the situation $R_1 \subseteq R_2$. In this case, we relate the existence of coordinated motion to the existence of a path on a graph constructed according to the configurations R_1 and R_2. Let V_1 and V_2 denote the set of vertices of the R_1 and R_2 boundaries. For each vertex i of V_2, we define

$L_i = \{ x \text{ in } V_1 \ / \ E_d(i,x) \geq r_1 + r_2 \} \cup \{x \text{ in } (a,b) \ / \ E_d(i,x) = r_1 + r_2 \text{ and } (a,b) \text{ is an edge of } R_1\}$ We define the vertex set V for our graph to be $V = \cup \{ (i,x) \ / \ x \text{ in } L_i \}$ We also construct points (I_1', F_1') and (I_2', F_2') on R_1 and R_2 corresponding to (I_1, F_1) and (I_2, F_2) and introduce the pairs (I_1', I_2'), (F_1', F_2') as vertices of G. Consider the line segment joining the points I_1, I_2 and let I_1', I_2' denote the furthest pair of points where this line intersects R_1 and R_2. Similarly construct F_1' and F_2'. Note that (I_1', I_2'), (F_1', F_2') are vertices in G. Based on Hausdroff distance notion, we define the edge set E of the graph G as follows : E_1 : There is an edge between $\{(i,x), (i,y)\}$ if x and y are adjacent vertices in R_1 and $H_d(i, (x,y)) \geq r_1 + r_2$. E_2 : There is an edge between (i,x) and (j,x) if $H_d(x, (i,j)) \geq$

$r_1 + r_2$. E_3 : Consider vertices (i,x) and (j,y) where i and j are adjacent in R_2 and x and y are adjacent in R_1. Let s_1 = (i,j) and s_2 = (x,y). This gives raise to 24 different configurations for s_1 and s_2. These are :

1. s_1 is a straight edge and s_2 is a circular arc and it is concave up. 1.1 s_1 and s_2 do not overlap. 1.2 s_1 and s_2 overlap. 1.3 s_2 is covered by s_1 fully. 1.4 s_1 is covered by s_2.

2. s_1 is a straight edge, s_2 is a circular arc and it is convex up. 2.1 s_1 and s_2 do not overlap. 2.2 s_1 and s_2 overlap. 2.3 s_2 is covered by s_1. 2.4 s_1 is covered by s_2.

3. s_1 is a circular arc (always convex up) s_2 is a circular arc and it is concave up. 3.1 s_1 and s_2 do not overlap. 3.2 s_1 and s_2 overlap. 3.3 s_2 is covered by s_1. 3.4 s_1 is covered by s_2.

4. s_1 is a circular arc, s_2 is a circular arc and it is convex up. 4.1 s_1 and s_2 do not overlap. 4.2 s_1 and s_2 overlap. 4.3 s_2 is covered by s_1. 4.4 s_1 is covered by s_2.

5. s_1 is a circular arc and s_2 is a straight edge. 5.1 s_1 and s_2 do not overlap. 5.2 s_1 and s_2 overlap. 5.3 s_2 is covered by s_1. 5.4 s_1 is covered by s_2.

6. s_1 and s_2 are straight edges. 6.1 s_1 and s_2 do not overlap. 6.2 s_1 and s_2 overlap. 6.3 s_2 is covered by s_1. 6.4 s_1 is covered by s_2.

In all these cases, we define edge between s_1 and s_2 as follows : If s_1 and s_2 do not belong to any of the cases 2.3, 4.1, 4.2, 4.3, 4.4 then there is a motion from (i,x) to

(j,y) if and only if the following condition is true : a. Either $H_d(x,(i,j)) \geq r_1 + r_2$ and $H_d(j,(x,y)) \geq r_1 + r_2$ or b. $H_d(i,(x,y)) \geq r_1 + r_2$ and $H_d(y,(i,j)) \geq r_1 + r_2$. Suppose (a) is true then by E_1 and E_2 edges would have been introduced between the vertices ((i,x), (j,x)) and ((j,x),(j,y)). A similar reasoning holds for (b) also. Hence, we do not have to introduce new edges in these cases. However, if s_1 and s_2 are in one of the cases 2.3, 4.1, 4.2, 4.3, 4.4 and if $H(s_1,s_2) \geq r_1+r_2$ then we introduce an edge between s_1 and s_2.

Note that it is sufficient to prove the existence or non-existence of a motion along the boundaries of R_1 and R_2. We shall show that a motion along the boundaries from (I_1',I_2') to (F_1',F_2') exists if and only if there is a path in G between (I_1',I_2') and (F_1',F_2'). Clearly this would confirm the motion of disks from (I_1,I_2) to (F_1,F_2).

Theorem 3 : There is a motion for the disks from (I_1',I_2') to (F_1',F_2') if and only if there is a path in G between the vertices (I_1',I_2') and (F_1',F_2').

Proof : Suppose there is a path in G between (I_1',I_2') and (F_1',F_2'). Then as a direct consequence of our definitions on edges, the existence of a motion for the disks follows. So, it remains to prove the converse.

Assume that there is a motion for the disks from (I_1',I_2') to (F_1',F_2'). We shall show that there is a path in G between (I_1',I_2') and (F_1',F_2'). We state and prove several lemmas leading to the final result.

Lemma 1 : There is a motion from (I_1', I_2') to (F_1', F_2') if and only if there is a motion along the boundaries R_1 and R_2 from (I_1', I_2') to (F_1', F_2').

Proof : The if-part is obvious. We prove the only-if part. Suppose there is a motion for the two disks from (I_1', I_2') to (F_1', F_2'). We have to prove that correspondingly there is a motion along the boundary for the two disks. Let C_1 and C_2 be the corresponding positions of the two disks on the trajectories at any instance.

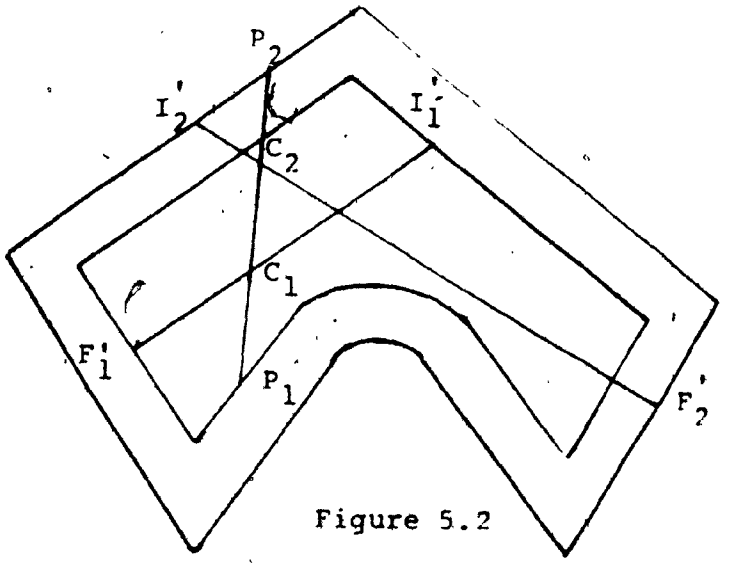


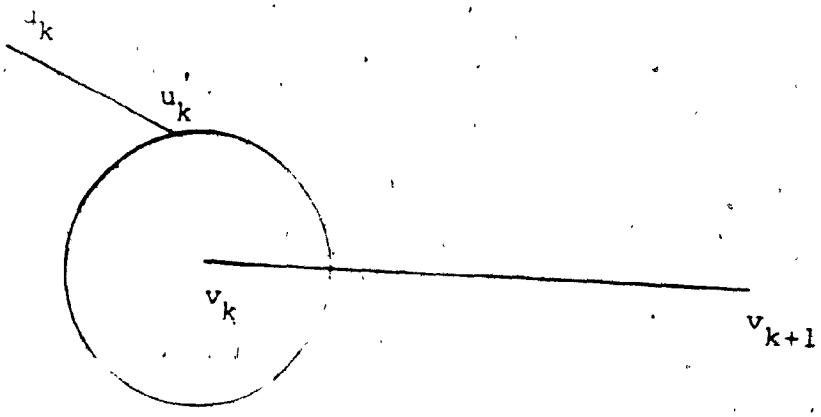
Figure 5.2

Let the line segment C_1C_2 meet the boundary of R_2 externally at P_2 and the line segment C_2C_1 meet R_1 externally at P_1. Obviously, $E_d(P_1, P_2) \geq r_1 + r_2$. Hence, we get a set of free positions on the boundaries which defines a motion along the boundary.

Lemma 2 : Any motion along the boundary can be transformed into one along the boundary that involves only the vertices of the boundary.

Proof : Although the motion is continuous, we can consider any motion as given by a sequence of points, say, (u_0,v_0), (u_1,v_1) .. (u_n,v_n), (u,v). It is implied that from (u_i,v_i) to (u_{i+1},v_{i+1}) the motion is along the edge as defined by these points. Moreover, it is continuous in the closed edge $<(u_i,v_i), (u_{i+1},v_{i+1})>$. Let (u_k,v_k) be the first pair such that either u_k or v_k (or both) is not a vertex of the boundary. (u_l,v_l), $l = 1, .. k-1$ is a pair of vertices of the region boundary R_1 and R_2. Let B_1 and B_2 denote the boundaries of the region R_1 and R_2.

Case 1 : u_k is not a vertex of B_1 and v_k is a vertex of B_2.



In this case, since u_k is not a vertex of B_1, it is possible to move C_1 at u_k to the nearest vertex u_k' of B_1, keeping C_2 at v_k. Hence, the pair (u_k,v_k) can be replaced by (u_k',v_k).

Case 2 : u_k is a vertex of B_1 and v_k is not a vertex of B_2. Let (v_{k-1}',v_k') be the edge on which v_k lies. If it is possible to move C_2 at v_k to v_k', keeping C_1 at u_k, then the pair (u_k, v_k) can be replaced by the pair (u_k,v_k') where v_k' is a vertex of B_2.

case 2.1 : Suppose such a motion from v_k to v_k' is not

possible. This implies that the circle C center u_k and radius $r_1 + r_2$, intersects the segment (v_k, v_k'). This implies that the segment (v_{k-1}', v_k) is outside the circle C. Hence $E_d(v_{k-1}', u_k) > r_1 + r_2$ and $E_d(v_k, u_k) \geq r_1 + r_2$. This shows that the previous vertex u_{k-1}' is outside or on the circle at v_{k-1}' and edge (u_{k-1}', u_k) is completely outside the circle at v_{k-1}'. Hence, C_2 can be kept at v_{k-1}' and C_1 can be moved to u_k. Hence the pair (u_k, v_k) can be replaced by (u_k, v_{k-1}').

case 3 : Both u_k and v_k are not vertices of B_1 and B_2.

Let (u_{k-1}', u_k') be the edge on which u_k lies and (v_{k-1}', v_k') be the edge in which v_k lies.

Case 3.1 : Suppose it is possible to keep C_1 at u_k and move C_2 from v_k to v_k'. Now, the pair (u_k, v_k) can be replaced by the pair (u_k, v_k') where u_k alone is not a vertex of B_1. The rest follows from case 1.

case 3.2 : It is possible to keep C_2 at v_k and move C_1 from u_k to u_k'. In this case, the pair (u_k, v_k) can be replaced by the pair (u_k', v_k) where v_k alone is not a vertex of B_2. The rest of the details are as in case 2.

case 3.3 : It is not possible to move C_1 to u_k' keeping C_2 at v_k and it is not possible to move C_2 to v_k' keeping C_1 at u_k. This implies that the circle with center v_k and radius $r_1 + r_2$ intersects the segment (u_k, u_k') and the circle with center u_k and radius $r_1 + r_2$ intersects the segment (v_k, v_k'). But $E_d(u_k, v_k) \geq r_1 + r_2$ and the circle at v_k intersects segment (u_k, u_k') implies that the segment (u_{k-1}', u_k) is

outside the circle at v_k. Hence, $E_d(u_{k-1}', v_k) \geq r_1 + r_2$. Since $E_d(u_{k-1}', v_{k-1}') \geq r_1 + r_2$ also holds, the edge (u_{k-1}', u_k') is completely outside the circle at v_{k-1}'. Hence C_2 can be kept at v_{k-1}' and C_1 can be moved to u_k'. In other words, the pair (u_k, v_k) can be replaced by the pair (u_k', v_{k-1}') where both u_k' and v_{k-1}' are vertices of B_1 and B_2.

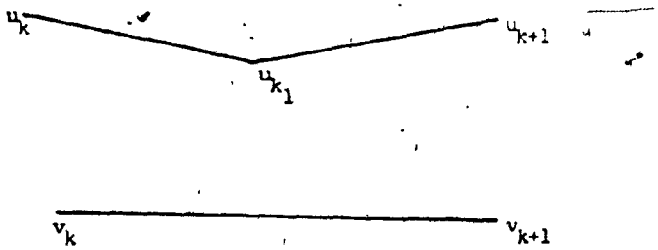
Hence we conclude that any motion along boundaries B_1 and B_2 can be replaced by a motion consisting only the vertices of the boundaries B_1 and B_2.

Lemma 3 : For every motion of the disks along the boundaries B_1 and B_2 involving only their vertices, there corresponds a path in the graph.

Proof : Let (u_0, v_0), (u_1, v_1) ... (u_∞, v_∞) be a motion along the boundary such that u_i and v_i are vertices of B_1 and B_2 respectively. We have to prove that this motion can be replaced by (u_0', v_0'), (u_1', v_1') ... (u_∞'', v_∞') where $((u_{k-1}', v_{k-1}'), (u_k', v_k'))$ is an edge of the graph. Consider any given pair (u_k, v_k) and (u_{k+1}, v_{k+1}).

Case 1 : Let u_k and u_{k+1} be adjacent in B_1 and v_k and v_{k+1} be adjacent in B_2. Let u_k, u_k^1, u_k^2 ... u_k^{n+1}, u_{k+1} be the sequence of adjacent vertices between u_k and u_{k+1}. We shall show that there is a path in G corresponding to the motion (u_k, v_k) to (u_{k+1}, v_{k+1}). We prove this by induction on the number of vertices between u_k and u_{k+1}.

Suppose there is only one vertex say u_k^1 in between u_k and u_{k+1}.



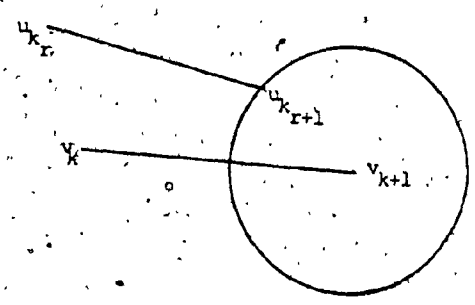
Let V be the position of C_2 when C_1 is at u_k^1. If $v = v_k$ then the path is $((u_k,v_k),(u_k^1,v_k))$ $((u_k^1,v_k),(u_{k+1},v_{k+1}))$. If $v = v_{k+1}$ then the path is $((u_k,v_k),(u_k^1,v_{k+1}))$ $((u_k^1,u_{k+1})(u_{k+1},v_{k+1}))$. If v is an interior point of the segment (v_k,v_{k+1}) then either the segment (v_k,v) is outside the circle of radius r_1+r_2 at u_k^1 or the segment (v,v_{k+1}) is outside the circle of radius r_1+r_2 at u_k^1. If (v_k,v) is free then $d(v_k,u_k^1) \geq r_1+r_2$ and $d(v_k,u_k) \geq r_1+r_2$. This implies that the edge (u_k,u_k^1) is outside the circle at v_k. Hence the given motion can be modified as (u_k,v_k), (u_k^1,v_k), (u_{k+1},v_{k+1}) and the corresponding path is $((u_k,v_k),(u_k^1,v_k))$ $((u_k^1,v_k),(u_{k+1},v_{k+1}))$. If (v,v_{k+1}) is free then the given motion can be modified as (u_k,v_k), (u_k^1,v_{k+1}), (u_{k+1},v_{k+1}) and the corresponding path is $((u_k,v_k),(u_k^1,v_{k+1}))$, $((u_k^1,v_{k+1}),(u_{k+1},v_{k+1}))$. This completes the proof for the basis of induction.

Next assume that the theorem is true when there are p (>1) vertices between u_k and u_{k+1}. We prove the theorem for the case of p+1 vertices between u_k and u_{k+1}. When the C_1 is at u_k^1, C_2 should be at some position v on the segment (v_k,v_{k+1}). If $v = v_k$ or $v = v_{k+1}$ then either $((u_k,v_k),(u_k^1,v_k))$ or $((u_k,v_k),(u_k^1,v_{k+1}))$ would be an edge in the constructed graph. If v is an interior point of the segment

(v_k, v_{k+1}) then either (1.1) the segment (v_k, v) is free or (1.2) the segment (v, v_{k+1}) is free. In case 1.1, the given motion can be modified to : fix C_2 at v_k and move C_1 to u_k^1. Correspondingly, $((u_k, v_k), (u_k^1, v_k))$ would be an edge in the graph. Now, (u_k, v_k), (u_{k+1}, v_{k+1}) is a motion in which there are n vertices in between u_k^1 and u_{k+1}. By hypothesis, this motion is recognizable as a path in the graph. In case 1.2, the given motion can be modified to : move C_1 from u_k to u_k^1 and C_2 from v_k to v_{k+1}. The corresponding edge in the graph is $((u_k, v_k), (u_k^1, v_{k+1}))$. Now we shall prove that the subsequent motion is recognizable by the graph.

Case 1.2.1 : The sequence of edges (u_k^1, u_k^2), (u_k^2, u_k^3) ... (u_k^{p+1}, u_{k+1}) are all outside the circle at v_{k+1}. In this case the path in the graph for the motion is $((u_k^1, v_{k+1}), (u_k^2, v_{k+1}))$, $((u_k^2, v_{k+1}), (u_k^3, v_{k+1}))$ .... $((u_k^{p+1}, v_{k+1}), (u_{k+1}, v_{k+1}))$.

Case 1.2.2 : There is a vertex u_k^r such that u_k^1, u_k^2... u_k^r are outside the circle at v_{k+1} and u_k^{r+1} is inside the circle.



This is a consequence of the result that any edge of R_1

should either be entirely outside the circle C center v_{k+1} and radius r_1+r_2 or entirely within it. It cannot penetrate the circle. Now, C_2 can be kept at v_{k+1} and C_1 can be moved upto u_k^r along successive edges and hence this motion is recognizable by the graph.

Case 1.2.2.1 : The edge (v_k,v_{k+1}) is outside the circle at u_k^1. In this case, C_2 can be moved to v_k keeping C_1 at u_k^r. By induction hypothesis, motion from (u_k^r,v_k) to (u_{k+1},v_{k+1}) is recognizable as a path in the graph since there are less than p vertices in between u_k^r and u_{k+1}.

Case 1.2.2.2 : The circle of radius r_1+r_2 at u_k^r intersects edge (v_k,v_{k+1}). We claim that there is a vertex u_k^i in the sequence u_k^1, u_k^2, ... u_k^r such that either C_2 can be moved to v_k keeping C_1 at u_k^i (iterative motion) or a simultaneous motion exists from (u_k^i,v_{k+1}) to (u_k^{i+1},v_k).

Suppose our claim is not true. Then the circle C_q at every point q in the sequence u_k^1, u_k^2... u_k^r intersects v_k, v_{k+1}. Further, our initial condition is that (v_k,v) is blocked. This gives that $\cap \{(v_k,v_{k+1}) \cap C_q\} \neq \phi$, $e = (u_k^i,u_k^{i+1})$, $1 \leq i \leq r-1$. Further $\{(v_k,v_{k+1}) \cap C_q\}$ has a continuous transition on (v_k,v_{k+1}) as q moves along (u_k^1,u_k^2), (u_k^2,u_k^3) ... continuously.

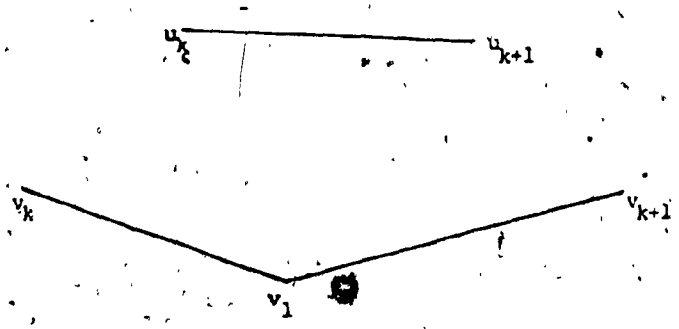
This shows that the corresponding position for C_2 in (v_k,v_{k+1}) when C_1 is at u_k^r is in the segment that includes v_{k+1}. This gives that there is no motion beyond (v_{k+1},u_k^r) which is a contradiction. Hence, our claim that C_2 can be moved to v_k through a simultaneous or iterative motion and

for this motion there is an edge in the graph. At this stage C_1 is at u_k^j, for some j, $1 \leq j \leq r$. Now, the forward motion from (u_k^j, v_k) to (u_{k+1}, v_{k+1}) is recognizable as a path in the graph due to induction hypothesis.

case2:

We assume that u_k and v_{k+1} are adjacent and v_k and v_{k+1} are not adjacent. We shall prove this case also using induction on the number of vertices between v_k and v_{k+1}.

Let there be only one vertex v_k^1 between v_k and v_{k+1}.



Let v be the position of C_1 when C_2 is at v_k^1. If $v = u_k$ or u_{k+1} then the motion is recognizable. Otherwise, $E_d(v, v_k^1) > r_1 + r_2$. The distance cannot be equal to $r_1 + r_2$ since in that case, v would be a vertex. This implies that the edge (u_k, u_{k+1}) is outside the circle at v_k^1. Hence, we can define the motion as (u_k, v_k) (u_{k+1}, v_k^1) (u_k, v_k^1) (u_{k+1}, v_{k+1}) and this is recognizable by the graph.

Next assume that when there are p (>1) vertices between v_k and v_{k+1} the motion is recognizable by the graph. We prove the case in which there are (p+1) vertices between v_k and v_{k+1}. Let v_k^1, v_k^2 ... v_k^{p+1} be the vertices. Let v

be the position of C_1 when C_2 is at v_k^1. If $v = u_k$ or $v = u_{k+1}$ then by hypothesis, the motion is recognizable. If v is an interior point of (v_k, v_{k+1}) then by our previous arguments, (u_k, u_{k+1}) is outside the circle at v_k^{+1}. Then the given motion can be redefined as : (u_k, v_k), (u_{k+1}, v_k^1), (u_k, v_k^1), (u_{k+1}, v_{k+1}). Due to the given motion, there are edges between (u_k, v_k), (u_{k+1}, v_k^1) and (u_{k+1}, v_k^1), (U_k, v_k^1). By induction hypothesis, motion from (u_k, v_k^1) to (u_{k+1}, v_{k+1}) is recognizable. Hence there is a corresponding path in the graph for the given motion.

Case 3 : Let there be s vertices u_k^1, u_k^2, ... u_k^s between u_k and u_{k+1} and t vertices v_k^1, v_k^2 ... v_k^t between v_k and v_{k+1}. Let u be the position of C_1 when C_2 is at v_k^1. If u is in (u_k^i, u_k^{i+1}) and is not a vertex, it can be moved to the nearest vertex u_k^{i+1}. Hence the motion is the sequence : $((u_k, v_k), (u_k^{i+1}, v_k^1))$ $((u_k^{i+1}, v_k^1)(u_k^i, v_k^1))$ $((u_k^i, v_k^1)(u_{k+1}, v_{k+1}))$. By case 1, motion from (u_k, v_k) to (u_k^i, v_k^1) is recognizable. In the same way, we can split the motion from (u_k^i, v_k^1) to (u_{k+1}, v_{k+1}) by considering the position of C_1 successively at v_k^2, v_k^3 ... v_k^{t+1}. In all these cases, we can use case 1 and derive an equivalent motion that is recognizable.

Hence we have shown that for an arbitrary k, $0 \leq k \leq \infty$, the motion from (u_k, v_k) to (u_{k+1}, v_{k+1}) is recognizable by the graph G. Since k is arbitrary we conclude that for the given motion there is a recognizable path in G. From lemmas 1, 2 , 3 we have shown that for any arbitrary motion of the

disks from (I_1', I_2') to (F_1', F_2') there is a corresponding path in G between (I_1', I_2') and (F_1', F_2'). This completes the proof of theorem 3.

Finally we give a brief analysis of our method to show that the cost of the entire algorithm is $O(n^2)$ where n is the total number of edges in the obstacles. It is easy to see that the cost of enlarging the boundaries and determining R_1 and R_2 is $O(n^2)$. Note that R_1 and R_2 can each have O(n) vertices. For each vertex i in R_2, $|Li|$ is O(n) and hence the vertex set V of G is $O(n^2)$. The number of edges introduced by types E_1 and E_2 is $O(n^2)$. Since there is O(n) circular edges in R_1 and R_2, the number of edges in type E_3 is also $O(n^2)$. Moreover Hausdorff distance between 2 edges (straight line segments or circular segments) can be computed in constant time. Thus the cost of constructing G=(V,E) with $|V| = O(n^2)$, $|E| = O(n^2)$ is $O(n^2)$. Now the path finding algorithm in [23] requires time $O(|V| + |E|)$ and thus the coordinated motion problem for two disks is solved in time $O(n^2)$.

Chapter 6

Coordinated motion of several disks

6.1 Introduction.

We give a generalization of our algorithm to determine the existence of a coordinated motion of several disks; when a motion exists our algorithm also constructs a collision free path. We first describe the generalization for k = 3 and then discuss the most general situation.

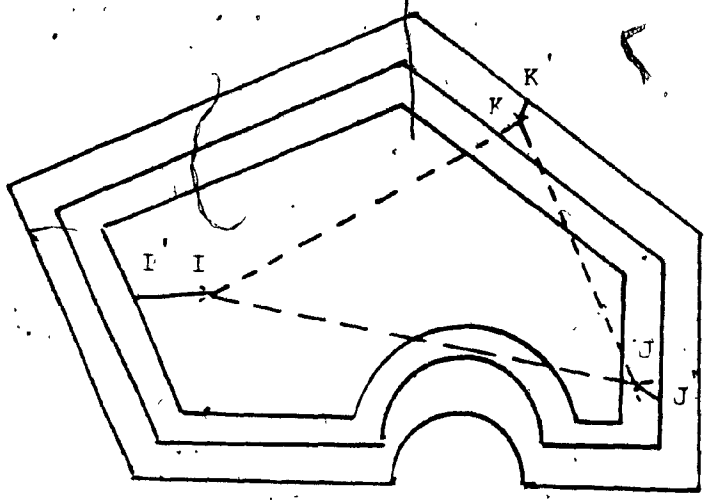
6.2 Coordinated motion of 3 disks

Let r_1, r_2 and r_3 be the radii of the three disks with $r_1 > r_2 > r_3$ and C_1, C_2 and C_3 be their centers. We displace independently the boundaries by r_1, r_2 and r_3 respectively and compute the sets G_1, G_2 and G_3 of r_1-components, r_2-components and r_3-components. In the next step, we identify the components R_1 in G_1, R_2 in G_2 and R_3 in G_3 to which (I_1,F_1), (I_2,F_2) and (I_3,F_3) belong. If any pair (I_i,F_i) belong to two different components then there is no motion. If R_1, R_2 and R_3 are pairwise disjoint, then the problem degenerates into single disk problems of moving C_3 in R_3, C_2 in R_2 and C_1 in R_1 and there is always a motion. if $R_1 \cap R_2 \neq \phi$ and $R_1 \cap R_2 \cap R_3 = \phi$ then we can prove $R_1 \subseteq R_2$ and $R_2 \cap R_3 = \phi$. Hence the given problem degenerates into the coordinated motion of C_1 and C_2 independent of C_3 and there is always a motion for C_3 from I_3 to F_3. Similar degenerate cases can also be considered in the same way.

Next we consider the case when the regions R_1, R_2 and

R_3 are not disjoint. It is straightforward to use theorem 1 of chapter 5 and prove that $R_1 \subseteq R_2 \subseteq R_3$. In this case, we define a graph G as in chapter 5 to determine the existence of a motion and a path in G if such a motion exists. The construction of G is briefly described below:

Let B_i, $i = 1,2,3$ denote the boundary of the region R_i. When R_i is multiply connected B_i is the union of several boundaries whose interior is R_i. We also let B_i denote the vertices on its boundary. From the context it should be obvious as to which we refer. The set of vertices include for example (I_1',F_1'), (I_2',F_2'), (I_3',F_3') which are constructed from (I_1,F_1), (I_2,F_2), (I_3,F_3) by a construction shown below: For any triple (i,j,k) of free positions, the corresponding points on the boundaries are (i',j',k') which are the points of intersection of the external bisectors of the angles at i,j,k with R_1,R_2 and R_3 boundaries. See figure.



If i,j,k are collinear, we perturb one of the points

slightly and then apply the above construction. For each i in B_3, we define

$L_i' = \{j \text{ in } B_2 \;/\; E_d(i,j) \geq r_3+r_2\} \cup \{j \text{ in } R_2\text{-boundary} \;/\; E_d(j,i) = r_2+r_3\}$;

$L_i'' = \{k \text{ in } B_1 \;/\; E_d(i,k) \geq r_1+r_3\} \cup \{k \text{ in } R_1\text{-boundary} \;/\; E_d(i,k) = r_1+r_3\}$.

For each j in B_2, we define

$L_j''' = \{k \text{ in } B_1 \;/\; E_d(j,k) \geq r_1+r_2\} \cup \{k \text{ in } R_1\text{-boundary} \;/\; E_d(i,j) = r_1+r_2\}$.

The vertex V of the graph is defined as follows:

$V = \cup \{(i,j,k) \;/\; j \text{ in } L_i',\ k \text{ in } L_i'' \cap L_j'''\}$. We also include the triads (I_3', I_2', I_1') and (F_3', F_2', F_1') in V.

The edge set is defined as follows:

Type 1: There is an edge between (i,j,k_1) and (i,j,k_2) if and only if k_1 and k_2 are adjacent vertices in R_1 and $H_d(j,(k_1,k_2)) \geq r_1+r_2$ and $H_d(i,(k_1,k_2)) \geq r_1+r_3$.

Type 2: Define $\{(i,j_1,k), (i,j_2,k)\}$ as an edge if and only if j_1 and j_2 are adjacent in R_2 and $H_d(i, (j_1,j_2)) \geq r_2+r_3$ and $H_d(k, (j_1,j_2)) \geq r_1+r_2$.

Type 3: Introduce $\{(i_1,j,k), (I_2,j,k)\}$ as an edge if and only if i_1, i_2 are adjacent in R_3 and $H_d(j, (i_1,i_2)) \geq r_2+r_3$ and $H_d(k, (i_1+i_2)) \geq r_1+r_3$.

Type 4: Let $\{(i,j_1,k_1), (i,j_2,k_2)\}$ be an edge if and only if $H_d(i, (j_1,j_2)) \geq r_2+r_3$ and $H_d(i, (k_1,k_2)) \geq r_3+r_1$ and edges (j_1,j_2) and (k_1,k_2) are one of the forms 2.1, 4.1, 4.2, 4.3, 4.4 (see chapter 5) and $H((j_1,j_2), (k_1,k_2)) \geq r_2+r_3$.

Similarly we define type 5 edges of the form $\{(i_1,j,k_1)$,

(i_2,j,k_2) and type 5 edges of the form $\{(i_1,j_1,k), (i_2,j_2,k)\}$.

Type 7: There is an edge between (i_1,j_1,k_1) and (i_2,j_2,k_2) if and only if edges (i_1,i_2), (j_1,j_2) and (k_1,k_2) are pairwise of the form 2.1, 4.1, 4.2, 4.3, 4.4 (see chapter 5) and $H((i_1,i_2), (j_1,j_2)) \geq r_3+r_2$, $H((j_1,j_2), (k_1,k_2)) \geq r_1+r_2$, $H((k_1,k_2), (i_1,i_2)) \geq r_3+r_1$.

It is obvious that all possible iterative motions are covered by edges of the first 5 types. Any motion for the three disks which is not given by these edges is captured by type 7 edges. Having defined the graph $G = (V,E)$, we can prove that there is a motion for the 3 disks from the initial configuration (I_1,I_2,I_3) to the final configuration (F_1,F_2,F_3) if and only if there is a path in G between (I_1',I_2',I_3') and (F_1',F_2',F_3'). We generalize and state below the most important theorem of chapter 5.

Theorem 4 : There is a motion for the disks from (I_1',I_2',I_3') to (F_1',F_2',F_3') if and only if there is a path in G between the vertices (I_1',I_2',I_3') and (F_1',F_2',F_3').

Proof : The proof is similar to the proof of theorem 3 and requires the associated lemmas also. We omit the details.

Although we can compute the connected components bounded by B_1,B_2, B_3 in time $O(n^2)$, note that there are $O(n^3)$ vertices and edges in G. Hence the cost of constructing G is $O(n^3)$ and the cost of determining the existence of a path in G is also $O(n^3)$.

5.3 Coordinated motion of k (> 3) disks.

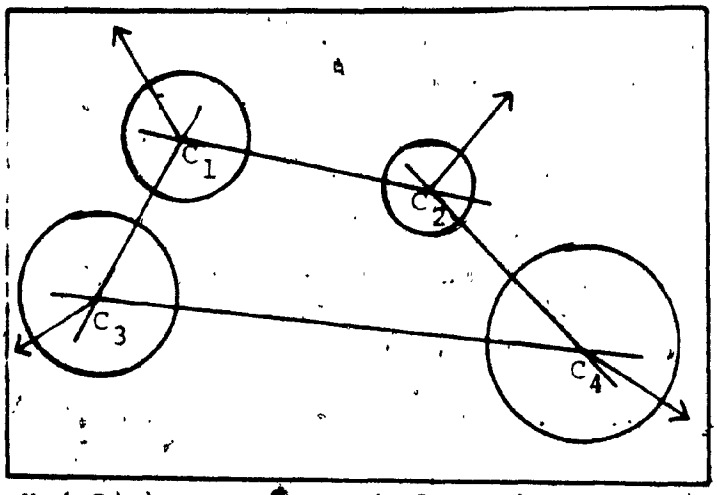
We give a few examples and state some results to motivate the generalized algorithm. First consider cases when two of the regions R_i, R_j , $1 \leq i \neq j < k$ are disjoint. This will reduce the problem into two subproblems and within each we deal with fewer than k disks. Since at this stage we assume that motion coordination for k-1 disks has been completely solved, we solve each one of these subproblems to obtain the solution to the original problem. Hence it is sufficient to consider the most general situations when $R_1 \subset R_2 \subset R3 \ , \ \ldots \subset R_k$.

Suppose $I = (I_1, I_2, \ \ldots \ I_k)$ and $F = (F_1, F_2, \ \ldots \ , F_k)$ are the initial and final positions of the disks. In order to generalize our algorithm to $k \geq 4$ disks, it is essential first to obtain $I' = (I_1', I_2', \ \ldots \ , I_k')$ and $F' = (F_1', F_2', \ \ldots \ , F_k')$ where (I_j', F_j') belongs to B_j, $1 \leq j \leq k$ and I', F' are free positions for the disks. This requires defining I' and F' in such a way that a coordinated motion of the disks from I to I' and F' to F are obvious or are easily derivable. For $k \geq 4$ such mappings are not always easy to define. See for example the figures below :

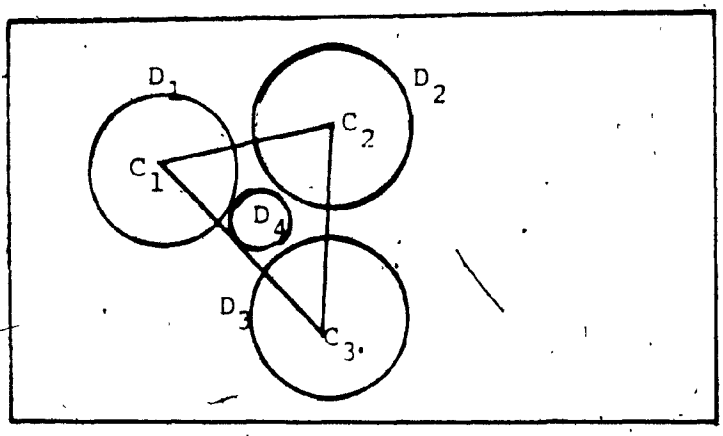
In figure 5.1, k = 4 and disks can be pushed to their boundaries along the directions shown.

In figure 5.2, k = 4 and D_4 cannot be pushed out without hitting the other disks.

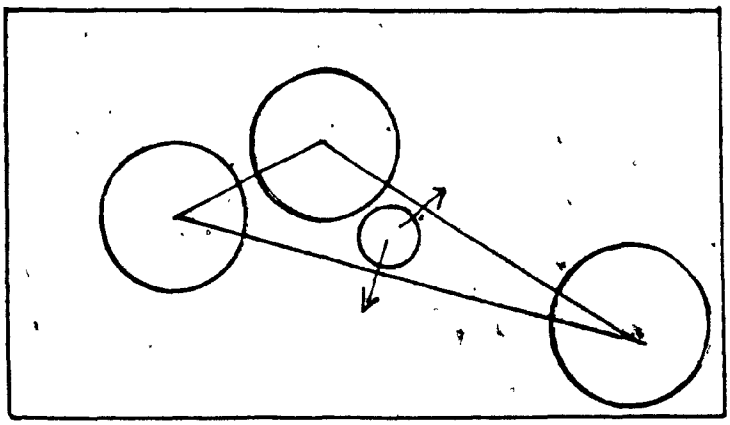
In figure 5.3, k = 4 and D_1 is first pushed to its boundary. Disk4 can be pushed in one of the directions shown.



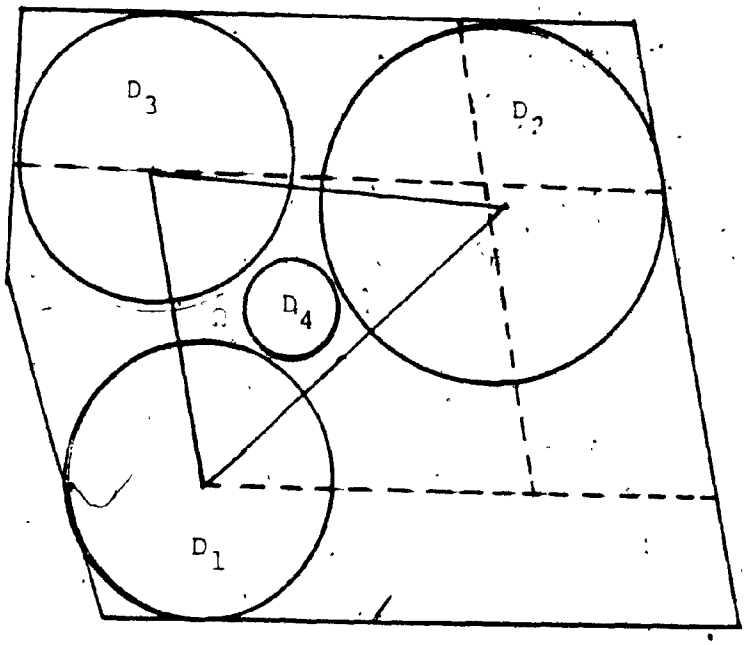
K=4:Disks can be pushed to their boundaries along the directions shown.



D_4 cannot be pushed out without hitting the other disks.



D_3 is first pushed out to its outer boundary. Now disk D_4 can be pushed in one of the directions shown.



D_4 cannot be pushed to its boundary.

From the example in figure we see that a motion for the four disks does not exist if the final position F_4 of the disk D_4 lies outside the region Ω. However, we cannot conclude the existence of a coordinated motion for the disks when F_4 lies inside Ω. Moreover, the present version of our algorithm does not directly apply to this situation. Thus we shall separately deal with the two issues, namely, defining the connectivity graph G for a given pair (I', F') of free positions on the boundaries and defining G when not all disks can be pushed to their respective outer boundaries.

Once we are given $I' = (I_1', I_2', \ldots, I_k')$ and $F' = (F_1', F_2', \ldots, F_k')$ with $I_j, F_j \in B_j$ and these are free

positions for the k disks. We construct G by defining V and E similar to our definitions in the last section. Informally stated, V is the disjoint union of several k-tuples of vertices of the boundaries, some old and some newly introduced so that each k-tuple is a free position for the disks. Such a set V can be constructed as follows :

For each i in B_q, define the set

$L_i(q,t) = \{j \text{ in } B_t \ / \ E_d(i,j) \geq r_q+r_t \text{ and } j \text{ is a vertex} \}$
$\cup \{j \text{ is not a vertex and } j \text{ in } B_t \ / \ E_d(i,j) = r_q+r_t\}$ for all $q, t : 2 < q \leq k, q+1 \leq t \leq k-1$.

Now define the vertex set V of G as :

$V = \{(i_1,i_2, \ldots ,i_k) \ / \ i_k \text{ in } B_k, i_{k-1} \text{ in } L_i(k,k-1), i_{k-2} \text{ in } (L_i(k,k-2) \cap L_i(k-1,k-2)) \ \ldots \ (i_1 \text{ in } \cap L_i(j,1))\}$

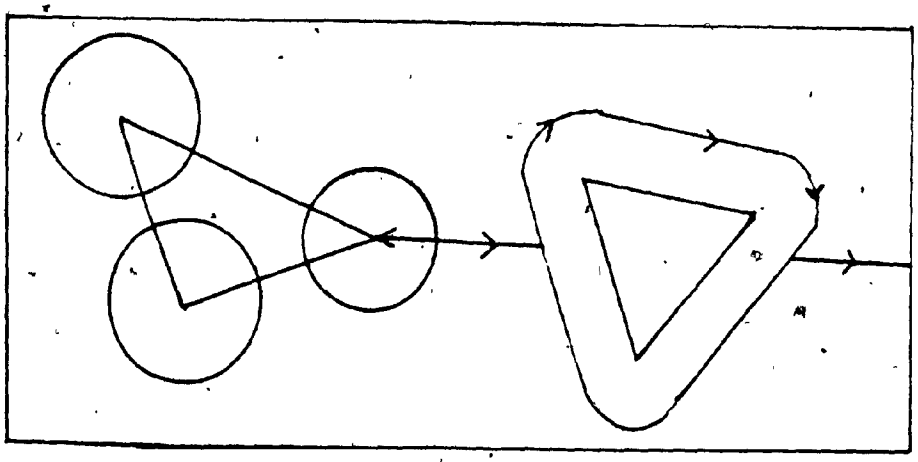
Next we define the edge set E of G. Informally, these are pairs of k-tuples which are adjacent on their boundaries so that a simultaneous motion of the disks along these edges is possible. We consider all pairs <s, t> where

$s = (i_1,i_2, \ldots ,i_k)$

$t = (j_1,j_2, \ldots ,j_k)$ where $i_p \neq j_p$, $i_q = j_q$ for $1 \leq q \neq p < k$. There is an edge in G between the vertices s and t if $H_d(i_q,(i_p,j_p)) \geq r_p+r_q$, $1 \leq q \neq p \leq k$. Suppose p out of k components of a tuple are the same and the remaining k-p are distinct. Without loss or generality assume $i_q \neq j_q$, $1 \leq q \leq p$ and $i_q \neq j_q$ for $p+1 \leq q \leq k$. Now we define an edge between the vertices $s = (i_1,i_2, \ldots ,i_k)$ and $t = (i_1,i_2, \ldots ,i_p,j_{p+1}, \ldots ,j_k)$ if $H_d(i_q,(i_t,j_t)) \geq r_q+r_t$ for $q = 1, 2, \ldots p$ and $t = p+1, \ldots , k$ and $H((i_q,j_q),(i_t,j_t)) \geq$

r_q+r_t for all q, t with $p+1 \leq q, t \leq k$.

Next we discuss a method to obtain (I',F') from (I,F) and develop methods to determine the existence of a coordinated motion. First consider the situation when the points $I_1, I_2, \ldots, I_k$ are vertices of a convex polygon. Now we consider the bisectors of the external angles at each I_j. It is easy to see that disk D_j can be pushed along its external bisector to its outer boundary. When this is done simultaneously, the disks cannot collide with each other. If at any instant of its motion, a disk encounters a boundary of an inner obstacle, it can move touching this wall until reaching the other end of the bisector (on the opposite side of the obstacle boundary) and then can be pushed along the bisector. See figure below :



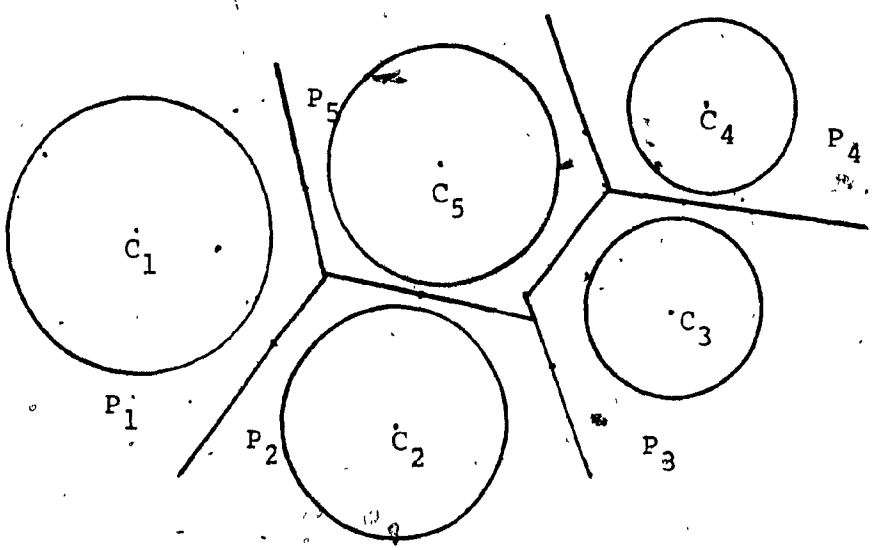
Next we consider the initial configuration I when the centers $I_1, I_2, \ldots I_k$ do not form the vertices of a convex polygon. We try to push as many disks as possible without

collision to their respective boundaries. One method to achieve this is described below :

Form the convex hull H_1 of the k points $I_1, I_2, \ldots I_k$. Now consider those centers which lie within H_1. Form H_2, their convex hull. Next we try to enlarge the convex hull H_1 as much as possible by pushing out without collision, one or more disks at the vertices of H_2. Without loss of generality we can assume the vertices of H_1 to be already on their respective outer boundaries. For each disk D_j on H_2, we find the nearest side of H_1. If its length is such that D_j can go through it without collision then push D_j out of H_1. Having done this for all the disks at the vertices of H_2 we enlarge H_1 with the new vertices (of the pushed out disks) and recompute H_2. We repeat this process until we cannot push out any more disk through a side of the recently formed H_1 or all the disks have been pushed out so that the disks are on their respective outer boundaries. In the latter case we are done. So we consider next the situation when a number of disks are on their respective outer boundaries with their centers forming the vertices of H_1 and H_1 has several disks in its interior. Such a configuration is collision free.

It may still be possible to push out some more disks provided the disks already on their outer boundaries are nicely packed. Nice packings might create an inter disk gap large enough to let several disks move out of H_1. Such a packing is done using a generalized Voronoi polygon

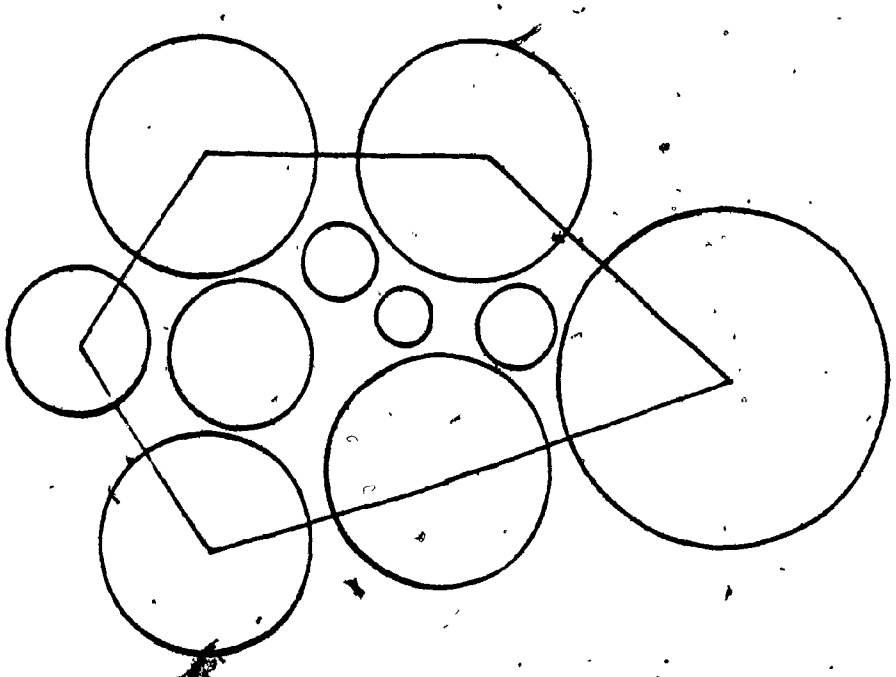
construction. For any two disks we define the distance between them to be the difference $\alpha - \beta$ where α is the distance between its centers and β is the sum of their radii. Consider the perpendicular line which bisects the distance between the disks. This line separates the disks in such a way that for all points within the half plane containing a disk D_1 this disk D_1 is nearer than D_2 which lies in the other half plane. When we have a number of disks, perpendicular lines for every pair of disks should be considered. These divide the entire region containing the disks into polygonal cells with each polygonal cell containing exactly one disk. See diagram below :



Note that a disk within its generalized Voronoi polygonal region can be moved freely inside its polygon without colliding with other disks. Hence we choose a pair of disks (adjacent on their convex hull) whose distance is a maximum and we separate them as much as possible. This is

done by moving them away within their Voronoi polygons until each disk just touches a side of its polygon.

Now the distance between these disks has increased and if disks from inside can be pushed through this side then it will be done. When this cannot be done and some disks remain in the interior we recompute the Voronoi polygons and repeat until either all disks have been pushed out to their outer boundaries or no more packing is possible. In the former case we are done and in the latter case we show below how a collision free motion can be determined, when one exists.

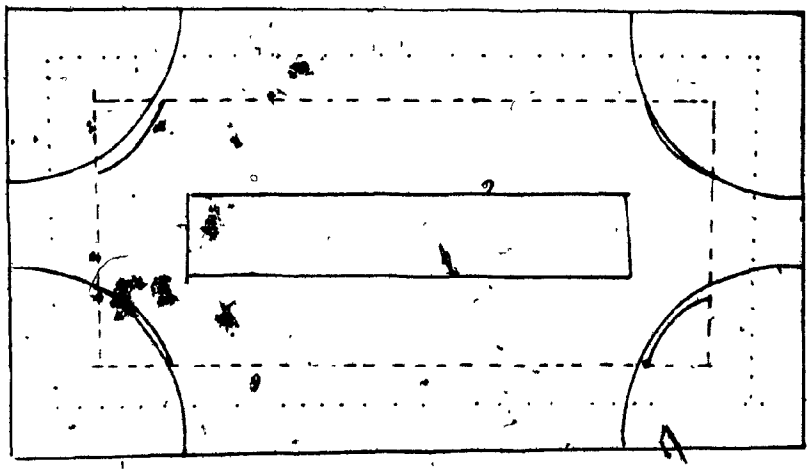


The diagram above shows a packed critical configuration with several disks in the interior which cannot be pushed to the boundaries. For definiteness, we assume that j ($\geq$ 1) disks are in the interior and (k-j) disks are on their respective boundaries. We denote the interior of H_1 by H_1. Since none of the j disks in Ω H_1 can be pushed out

without colliding with a disk on H_1, we conclude that a coordinated motion is impossible when the final positions for some of the disks in ΩH_1 are in its complement region; (for example on H_1). So we are left to consider the only case when the final positions of disks in ΩH_1 belong to the open region ΩH_1.

We need a definition and two lemmas for our further discussion.

Definition : For any two circular disks of radii r_1 and r_2, the straight line segments which lie at distance r_1+2r_2 from a wall W and circular arcs which lie at distance r_1+2r_2 from a convex corner at which two walls meet are called critical curves. An open connected region enclosed by several critical curves is called a non-critical region for r_2 because this contains the set of positions at which the disk of radius r_1 can be placed safely when r_2 is placed touching an outer wall.



Enlarged boundaries and critical curves

Lemma : Consider three disks of radii $r_1 > r_2 > r_3$. Now, $r_1+2r_3 > r_2+2r_3$. A non-critical region for r_1 is contained within a non-critical region for r_2 when r_3 is fixed touching an outer wall.

Proof : Since every free position of r_1 is a free position of r_2 and non-critical regions are subsets of free positions the result follows.

Lemma [Schwartz and Sharir]:

For each noncritical region R, $\sigma(R) = \sigma(x)$ where x is any arbitrary point of R and $\sigma(x)$ is the set of all connected components of free positions available to r_1 when r_2 is placed at x. $(r_1 > r_2)$

This lemma asserts that for a noncritical region R (say with respect to a disk r_2) the disk r_2 can be considered as a static obstacle for determining the motion of the other disk r_1.

Now consider the disk, say D_i, of H_1 of maximum radius. This disk lies on B_i, the r_i-enlarged boundary. We consider each disk in the interior of H_1 and determine the noncritical regions with D_i fixed. Below we shall explain our method.

Choose a disk say D' from the interior of H_1 and enlarge B_i by an amount equal to the diameter of the chosen disk. The straight line and circular segments are the critical curves and these divide the region R_i into a number of connected components. (Recall our descriptions in chapter 4). These are exactly the noncritical regions with

respect to D_i and the chosen disk D'. Now the center of the disk D' belongs to one of these noncritical regions. We find this and call this σ_i. Since for any other disk D_t on H_1 we have $R_i \subset R_t$, by lemma $\sigma_i \subset \sigma_t$. Hence σ_i is the smallest noncritical region containing the center of D'. Now we repeat this process for D_i and every other disk in the interior of H_1. We have thus determined at most j noncritical regions containing the centers of the disks in $\Omega(H_1)$.

Next we determine for each pair (I_p, F_p) of initial and final positions of a disk D_p the noncritical region to which F_p belongs. If I_p and F_p belong to two different noncritical regions then the line segment I_pF_p can intersect a maximum of j noncritical regions. We determine these points and each one of these is a possible position for D_p in which it can be placed safely while all disks on H_1 can be moved without collision. Note that when I_p and F_p belong to the same noncritical region, the disk D_p can be moved from I_p to F_p without colliding with any other disk. Because there are j disks in the interior of H_1 and there are at most $O(j^j)$ j-tuples of points to be considered for safe placements of interior disks. Let S denote this set of j-tuples.

Next we explain how a suitable subset of these j-tuples can be chosen to adjoin with (k-j) positions of disks on the outer boundaries in order to form V, the vertex set of G.

Clearly any j-tuple $(A_1, A_2, \ldots, A_j)$ where each A_i

corresponds to a possible placement of an interior disk is a feasible position for the entire coordinated motion if the j interior disks when placed at these points do not collide with each other. Thus we eliminate from S those tuples which do not satisfy this safe placement property for the j interior disks.

Define the vertex set W from the outer boundaries on which the k-j disks on H_1 reside. This is done using our earlier definitions appropriately modified. Each vertex in W is a k-j tuple (representing a collision free placement of the k-j disks on their boundaries). Each j-tuple in S represents a collision free placement of j interior disks whenever the k-j disks are restricted to be in their respective outer boundaries. Thus we expand tuples of W by appropriately inserting vectors from S in all possible ways to obtain V. From our construction it is clear that V contains k-tuples of free positions for the k disks. Moreover from the lemma which characterizes the invariance of noncritical regions it follows that all possible free positions of the j interior disks are captured in the definition of V. The definition of E, the edge set of G is identical to our definition .

Now, a coordinated motion for the disks can be shown to exist if and only if there is a path in G between $I' = (I(H_1), I(\Omega H_1))$ to $F' = (F(H_1), F(\Omega H_1))$ where $(I(H_1), F(H_1)$ are the initial and final positions of the k-j disks on H_1) and $(I(\Omega H_1), F(\Omega H_1))$ are the initial and final positions

of the j disks in H_1. It is clear that when there is a path in G there is a collision free coordinated motion for the disks along the edges corresponding to the path in G. The proof of the converse follows from two observations : 1. The lemma [Schwartz , Sharir] enables us to construct the maximal noncritical region common to all the disks on H_1. Hence if the disks in ΩH_1 are placed in collision free positions and are moved not colliding with each other, there is a collision free coordinated motion for all the disks only when those disks on H_1 can be moved in a collision free manner along the boundaries. 2. The disks on H_1 cannot collide with other moving disks belonging to their common critical region. Moreover there is a collision free motion for them if and only if there is a path between $(I(H_1), F(H_1))$ in the graph constructed only from their vertices. Because this graph can be obtained from G by a projection to a lower k-j dimensional set of vertices, the existence of a coordinate motion implies that both G and a corresponding path in G can be constructed.

A more formal proof of this claim will require only lengthy descriptions without contributing any new insight. So we shall not attempt it and instead analyze the complexity of this generalized algorithm.

A careful review of our method reveals that the maximum amount of work is done when we have j out of k disks in the interior, push some of them to their boundaries, determine noncritical regions, form the graph G and determine a path

in G. Observe that $|V| = O(n^k)$ because there are atmost $O(n^{k-j})$ tuples in W and $O(j^j)$ tuples in S and hence there are atmost $O(j^j n^{k-j})$ tuples in V. Since $j < k$ and k is fixed $j^j n^{k-j} < n^k$. So, $|V| = O(n^k)$. It is evident that $|E| = O(n^k)$. Hence the cost of finding a path between two specified vertices in G is $O(n^k)$.

The number of noncritical regions formed is $O(j)$. For each of them, we need to enlarge B_i by a specified amount. Since B_i has $O(n)$ edges, the total cost of enlargement is $O(jn) = O(n)$ since j ($< k$) is independent of n. The cost of finding the connected components cannot exceed $O(n^2)$, since atmost all pairs of edges (after enlargement) must be examined. Hence the total cost of determining the noncritical regions containing the centers is $O(n^2)$.

The cost of determining the set S of feasible positions for the j interior disks is purely a function of j and hence must be considered a constant (independent of n).

Finally we must carefully evaluate the cost of finding the convex hulls H_1 and H_2 and successive enlargements. A convex hull of j points can be formed in time $O(j \log j)$. The generalized Voronoi polygons for the j disks can be constructed in $O(j^2 \log j)$ time. See Lee and Drysdale [9] for Voronoi diagram construction for circles. After each Voronoi construction we need to examine a maximum of j disks to choose a candidate disk and a corresponding side to be enlarged for pushing this out. This step can be done systematically in an exhaustive manner. The cost is a

function of j ($< k$) and hence is a constant (independent of n). It is fairly easy to see that this cost is a polynomial in j, the number of interior disks.

Putting these together, we conclude that the entire cost of our motion planning algorithm is $O(n^k)$.

Chapter 7

Some Concluding Remarks

The most challenging aspect of robotic manipulators would be the design of efficient algorithms for realistic set of object-obstacle instances. However, the computational complexity results of Reif [18] and Hopcroft, Schwartz and Sharir [5] clearly showed that efficient algorithms for a large class of nontrivial problem instances cannot exist. The chief motivation of this thesis comes from these observations.

Motion planning and collision avoidance algorithms were first studied by Udupa [24] and Lozano-Perez and Wesley [12] and these contained no complexity results. This thesis investigates these approaches for a simple robotic model and analyzes the computational complexity. The most general algorithm proposed by Schwartz and Sharir [21] requires $O(n^e)$ where the exponent can be quite high (no bound was given). Yap [25] has given $O(n^2)$ and $O(n^3)$ algorithms to solve the coordinated motion of two and three disks. His method, based on a principle called retraction cannot be generalized for $k \geq 4$ disks. As far as we know, this thesis is the first attempt to develop an algorithm that is uniformly applicable for determining the existence of a collision free motion for k (≥ 2) disks. Our algorithm has time complexity $O(n^k)$ where n is the number of edges (both straight edges and circular) composing the boundaries of obstacles.

The algorithm developed in this thesis does a global analysis of the geometry and the topology of the given configuration and determines the set of connected components of the free positions. The algorithm separately computes a connected component R_j with $O(n)$ edges enclosing the initial position I_j and final position F_j. Having found this for all disks, our algorithm constructs a graph G defined over the product space $B_1 B_2 \ldots B_k$ where B_i is the boundary of R_i. Since the product space has $O(n^k)$ vertices and the edges of G are chosen satisfying the collision free property for every pair of disks, the graph G has $O(n^k)$ edges as well. It is essential to consider a mapping of $I = (I_1, I_2, \ldots, I_k)$ and $F = (F_1, F_2, \ldots, F_k)$ onto the outer boundaries so that the existence of a motion can be related to the existence of a path in G. For $k \leq 3$ such a mapping is easy to define. For $k \geq 4$ such a mapping may not exist or may not be easy to find even when one exists. The methods outlined in chapter 5 either finds a mapping or determines that such a mapping cannot exist. In either case, our methods show how G, a connectivity graph can be defined and the existence of a motion is related to the existence of a path in G. We believe our algorithm is optimal.

The representation of contours and bounded regions discussed in chapter 4 enables an $O(n)$ algorithm for solving the motion planning problem for a single disk. This representation is an adaptation of the one used by Merrill

[13]. Such a grid representation may not be valid in very general situations; however within the context of determining components bounded by straight edges or circular edges, the inherent approximation of representation does not affect the accuracy demanded in identifying the connected components. In other words the connected components are always determined accurately thus solving the motion planning problem for one disk in time $O(n)$. Hence our algorithms in this thesis solve the motion planning problem of $k \geq 1$ disks in time $O(n^k)$.

Finally we remark on the generality of our method. Having determined the existence of motion for a given set of obstacles, if one or more obstacles is introduced, our method can be extended with minimum cost to determine the existence of motion in the new configuration. Our method can also handle the situation when one more disk is introduced after a motion for k disks have been determined. However, if I and F are not known in advance then all connected components must be saved until I and F become known. This requires only more storage and causes no change in the overall time complexity.

When an obstacle is deleted from consideration, the resulting set of free positions and the set of connected components will change. A similar situation would arise when a disk of different radius is given. In these situations, our algorithm can be modified; but we are not certain about minimal modifications.

There are several motion planning problems in the plane not yet fully investigated. Some of these are 1. Proving formally the optimality of our algorithm. 2. Adaptation of our method for the motion of one or more ladders (thickness may be omitted for simplicity) 3. Complexity of coordinated motion planning for elliptical plates (thickness may be ignored) 4. Motion planning for a platform with n circular wheels.

REFERENCES

1. V. S. Alagar and G. Ramanathan, "Algorithmic motion planning in robotics: A simplified model and an efficient solution", Proc. IEEE International Conference on Computers, Systems and Signal Processing, pp.761-768, 1984.

2. R. A. Brooks, "Solving the find-path problem by representing free space as generalized cones", Artificial Intelligence Laboratory, MIT, May 1982.

3. R. A. Brooks and T. Lozano-Perez, "A subdivision algorithm in configuration space for find-path with rotation", Artificial Intelligence Laboratory, MIT, December 1982.

4. M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, San Francisco, 1979.

5. N. Hogan, "Impedence control of a robotic manipulator", Winter Ann. Meeting of the ASME, Washington, D. C., November 1981.

6. J. E. Hopcroft, J. T. Schwartz and M. Sharir, "On the complexity of motion planning for multiple independent objects: PSPACE Hardness of the Warehouseman's problem", Tech. Report 103, Courant Institute of Mathematical Sciences, N. Y., February 1984.

7. J. E. Hopcroft, D. Joseph, S. Whitesides, "On the movement of robot arms in 2-dimensional bounded regions", Technical Report, Cornell University, 1982 and in SIAM J. Computing, 1984. (to appear)

8. D. Kirkpatrick, "Efficient computation of continuous skeletons", 20th IEEE Symp. FOCS, pp.18-27,1979.

9. D. T. Lee and R. L. Drysdale, "Generalizations of the Voronoi diagrams in the plane", SIAM J. Computing, pp.73-87,1981.

10. R. A. Lewis, "Autonomous manipulation on a robot: summary of manipulator software functions", Jet Propulsion Laboratory, California Institute of Technology, March 1974.

11. T. Lozano-Perez, "Spatial planning: a configuration space approach", IEEE Trans. Computers, pp.108-120,1933.

12. T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision free paths amongst polyhedral obstacles", Comm. ACM-22, 10, pp.560-570,1979.

13. R. D. Merrill, "Representation of Contours and regions for efficient computer search", Comm. ACM 16, pp.69-82.1973.

14. C. O'Dunlaing and C. K. Yap, "The Voronoi diagram method for motion planning: The case of a disk:, ACM Symposium on Theory of Computing, pp.207-220,1983.

15. R. P. Paul, Robot manipulators: Mathematics, Programming and Control, MIT Press, 1981.

16. D. I. Pieper, "The kinematics of manipulators under computer control", Ph.D. Thesis, Standford University, 1968.

17. G. Ramanathan and V. S. Alagar, "Algorithmic motion planning in robotics: Coordinated motion of several disks amidst polygonal barriers", to appear in IEEE International Conference on Robotics and Automation, St. Louis, March 1985.

18. J. Rief, "Complexity of the mover's problems and generalizations", Proc. 20th IEEE Symposium on Foundations of Computer Science,pp.421-427,1979.

19. J. T. Schwartz and M. Sharir, "On the piano movers' problem: I. The special case of a rigid polygonal body moving amidst polygonal barriers", Comm. Pure Appl. Math. Vol. XXXVI, 345-398, 1983.

20. J. T. Schwartz and M. Sharir, "On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds", Adv. Appl. Math. 4, 298-351, 1983.

21. J. T. Schwartz and M. Sharir, "On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers", Courant Institute Technical Report 52, September 1982.

22. M. I. Shamos and D. Hoey, "Closest-points problems", Proc. 16th IEEE Symposium on Foundations of Computer Science, pp.151-162,1975.

23. R. Tarjan, "Depth-first search and linear graph algorithms", SIAM J. Computing,pp.147-159,1972.

24. J. S. Udupa, "Collision detection and avoidance in computer controlled manipulators", Ph.D. Thesis, California Institute of Technology, 1977.

25. C. Yap, "Coordinating the motion of several disks", Technical Report, Courant Institute of Mathematical Sciences, February 1984.