RSD-TR-27-86

Robot Path Planning Using Geodesic and Straight Line Segments with Voronoi Diagrams

by

Kang G. Shin

Robert D. Throne

Department of Electrical Engineering and Computer Science
Center for Research on Integrated Manufacturing
The University of Michigan
Ann Arbor, MI 48109

December 1986

The work reported here is supported in part by the NSF under Grant No. ECS-8409938 and the US AFOSR under Contract No. F33615-85-C-5105. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors do not necessarily reflect the view of the funding agencies.

† Henceforth the term "position" or "point" will be used to mean both position and orientation.

Abstract

We develop a method for determining a continuous path for a robot manipulator's tip from an initial position and orientation to a final point[†] in a room filled with convex polygon obstacles. First, suitable intermediate points are selected which, when connected with straight line segments, produce an obstacle avoiding path. These points are then linked with suitable path types (straight lines or geodesics). These path segments are joined with fifth order polynomials to generate all obstacle avoiding paths which can be constructed from the path segments. Finally, the minimum traversal time for each of the paths generated is computed to determine which of the paths generated has the smallest traversal time.

Simulation results indicate that whenever paths made up of geodesic segments pass through the same intermediate points as paths made up of straight line segments, the geodesic paths can be traversed in less time. However, a path composed of straight line segments may have the smallest traversal time of those paths generated since the corresponding geodesic path may intersect an obstacle.

TABLE OF CONTENTS

1 Introduction

Robots are increasingly being used in industrial applications because of their high potential for increased productivity and improved quality. Ideally, the robot should be able to avoid obstacles while not wasting valuable time. We develop a method of determining a continuous path for a robot manipulator's tip from an initial position to a final position in a room filled with convex polygon obstacles. Along this path the manipulator's tip must not intersect any of the obstacles and the traversal time should be as small as possible.

Lin, Chang, and Luh [1] investigated the use of a suitable number of prescribed or fixed knot points fitted with cubic splines to produce minimum time paths. They introduced two additional knot points which could be placed so as to produce the minimum time path through the prescribed knot points. Rajan [2] also presented a procedure based on cubic splines where all of the knot points were allowed to move to produce a minimum time path. However, it is not clear whether the path produced is a local or global minimum. In each of these papers obstacle avoidance can be achieved if a sufficient number of knot points are chosen in advance so that the cubic splines do not intersect any obstacles, though neither of them allows for the generation of new knot points that would be required if the cubic splines did intersect an obstacle. Also, in both papers, the path must connect all of the knot points, and is not allowed to skip a knot point even if a trajectory passing through each intermediate point requires a longer traversal time.

Sahar and Hollerbach [3] presented a method for determining a minimum time trajectory through the combined use of joint space tesselation, dynamic time scaling, and graph searching. Though good results appear to have been obtained with this method, even relatively coarse meshes require substantial computation.

Gilbert and Johnson [4] expressed obstacle avoidance in terms of distance functions, and used optimal control principles to derive differential equations whose solutions are the geometric paths which minimize a prescribed functional. This method also introduces computational and numerical difficulties.

If we know that a certain path type, e.g., a geodesic segment, is nearly optimal, then connecting intermediate points with this path type may produce a path (nearly) locally optimal. As the distance between intermediate points is increased, the path composed of these near optimal path segments may perform better than a path composed of splines or Cartesian straight line segments. For this reason, our paths are allowed to bypass intermediate points so long as the resulting paths do not intersect any obstacles. However, because we do not know the exact route a geodesic path segment will take between two intermediate points until we have constructed it, we must systematically construct and check all of the possible path segments connecting intermediate points to determine if they intersect any of the obstacles. While the overall path we obtain may not be a global optimum even if we know all of the locally optimal paths between intermediate points, we will arrive at a good suboptimal path.

In this paper, we use a Voronoi diagram to determine intermediate points initially so that a path consisting of straight line segments can be found to connect the initial and final points. Once these intermediate points have been determined, the path segments joining these points can be constructed by various techniques. In particular, we will join these intermediate points with both straight line segments and geodesics. Cartesian straight line paths have been extensively studied in [5] and [6]. For the collision-free case, Shin and McKay [7] introduced geodesics in inertia space as an approximate minimum time path, with the geodesic path an exact minimum time path under certain restrictive conditions. Because of the method used to construct the intermediate points, and because it is difficult *a priori* to determine the path of a geodesic curve segment, the resulting path need not pass through all of the intermediate points so long as it does not intersect any obstacles. However, since more than one path from the initial position to the final position may be possible, we would like to generate all possible obstacle avoiding paths using the previously generated path segments and determine which of these paths requires the smallest traversal time.

Once the intermediate points are generated and the path segments computed, the path segments must be joined to form a single continuous path. Various methods have been proposed for joining path segments. Kim and

Shin [8] performed local optimization at each corner (intermediate) point and utilize the dynamics of the manipulator to minimize the transition times between path segments. Paul [5] assumed a fixed transition time that is chosen to allow for velocity changes from minimum to maximum and from maximum to minimum. Luh and Lin [9] proposed a nonlinear programming method to choose the transition points along straight line path segments.

Because we will determine the minimum time trajectory for each of our paths using the method presented in [10], which requires the path to be continuous with continuous first and second derivatives, quintic polynomials will be used as the joining functions. Specifically, we will join each curve in joint space with a different quintic polynomial rather than joining the path segments in Cartesian space with a single quintic polynomial. Since geodesics are naturally functions of their arc length (s) in inertia space, these quintic polynomials will also be parameterized by their arc length in inertia space. Determining which of the sample points on two path segments are to be joined with the quintic joining functions would be very inefficient if all possible sample points on each segment were tested. Instead, we only look at a fixed number of sample points on each segment. After all of the joining paths between the chosen sample points are generated, we use the quintic polynomials which have the minimum average absolute value of curvature in inertia space. A geodesic in inertia space has zero curvature, hence in some sense our joining paths will be like a geodesic.

Finally we determine the minimum time path among those generated.

The remainder of this paper is organized as follows. Section 2 states the problem. Section 3 discusses the method used to solve the problem, as well as the major algorithms used in the solution. Section 4 presents typical simulation results for the first three joints of the Stanford manipulator, and Section 5 contains the conclusions.

2 Problem Statement

Given the two Cartesian points describing the position of a manipulator's tip, denoted by $P_{initial}$ and P_{final}, in a room filled with convex polygon obstacles, the problem is to determine a collision avoiding path which requires as small a traversal time as possible. Specifically, we first want to generate intermediate points P_i such that for a given curve type (i.e., straight line or geodesic) we can find at least one continuous path from $P_{initial}$ to P_{final}. For $P_{initial}$, P_{final}, and each intermediate point P_i, we then want to construct all obstacle avoiding path segments that originate at that point and end at each of the other points. All paths which can be constructed from these path segments must be determined whether or not the resulting path passes through all of the intermediate points. If more than one continuous path is found from $P_{initial}$ to P_{final}, we want to determine which of these paths will require the least traversal time.

Since there are $N(N-1)/2$ possible path segments for N intermediate points (including $P_{initial}$ and P_{final}), the algorithm used must determine all of the segments without storing unnecessary or duplicate information.

A simple analysis shows that for N intermediate points (including $P_{initial}$ and P_{final}) there are 2^{N-2} possible paths[1] from $P_{initial}$ to P_{final}. Due to the rapid growth of the number of possible paths, it is very important to generate and store only those segments which are needed.

3 Solution Method

The solution to the problem is partitioned into the following five steps:

- The selection of suitable intermediate points.
- Generating the path segments to connect each intermediate point with all of the other intermediate points. Those segments that intersect an obstacle are discarded. If any other intermediate points are needed for collision avoidance, they are generated during this phase of the solution procedure.
- Determining all possible obstacle avoiding paths from the initial position ($P_{initial}$) to the terminal position (P_{final}) which are constructed by joining the path segments which connect intermediate points.

[1]Assuming none of the paths intersected an obstacle. This is the maximum, and for most cases far fewer obstacle avoiding paths are actually possible.

- Joining the path segments together to form a single continuous path with continuous first and second derivatives.
- Determining the minimum time required to traverse each of the paths.

In the remainder of this section, the the first four of the above five steps are described in detail. The details of determining the minimum time trajectory for a suitably parameterized path are described in [10].

3.1 Intermediate Point Selection

The initial intermediate points are determined by generating the Voronoi diagram for the points that make up the corner points of the obstacles. Dúnlaing and Yap [11] studied the case of a circular disk moving among polygon obstacles and proved that if there exists a collision-free path from an initial position to a final position, then there exists a collision avoiding path along the *generalized* Voronoi diagram[2]. We do not use the generalized Voronoi diagrams since we are only interested in selecting suitable intermediate points which can be connected with Cartesian straight line segments and geodesic segments. However, if the obstacles are large compared to the workspace, then more than just the corner points must be used in generating the Voronoi diagram. For example, we may have to use eight points for each rectangular obstacle (the four corner points and the four midpoints of each side) instead of using just the four corner points. The algorithm used to generate the Voronoi diagram was similar to that presented in [12], which requires that at most three edges in the Voronoi diagram intersect at a single point. This requires a slight modification of the position of one end point of each of the rectangular obstacles used. since the local Voronoi diagram of the four corner points is made up of the bisectors of the outside edges (which all meet at the center of the rectangle). In some degenerate cases, we may have to slightly alter more than one of an obstacle's corner points to avoid having more than three edges of the Voronoi diagram intersecting at a point due to the effects of the other obstacles. If this causes a

[2]This does not take into account the initial and final portions of the path, when the disk must be moved to the Voronoi diagram.

problem a better implementation of the algorithm presented in [12], which removes this restriction, would be an alternative.

Depending on the manipulator structure and the general location of the initial and final positions, it may be necessary to insert some *pseudo obstacles.* These are different from those presented in [13] which were introduced to prevent the boom of the Stanford arm from colliding with any of the real obstacles. We use pseudo obstacles to force the Voronoi diagram to generate paths which remain within the manipulator's workspace. The center obstacle in Figure 1 is such a pseudo obstacle, and it forces the paths away from the manipulator (which is located at the center of the pseudo obstacle).

An important restriction on the method used to generate the intermediate points is that they all must lie in the same plane. Although we could have used a three dimensional Voronoi diagram to choose intermediate points, we were only interested initially in essentially planar problems and chose to use only the planar Voronoi diagram. However, though all of the intermediate points are in the same plane, the joining paths need not remain in the same plane. In fact, the geodesic path segments will not remain in the plane of the intermediate points.

Having generated the Voronoi diagram, suitable intermediate points can now be computed by determining the points where the edges that make up the diagram intersect. These intermediate points, the initial point, and the final point are arranged in the sequence that would occur if each point was included in the final loop-free path. The initial point is labeled as point 0 and the final point is labeled as point N, assuming that there are $N - 1$ intermediate points found from the Voronoi diagram.

3.2 Generation of Path Segments and New Intermediate Points

Each point is associated with a data structure or record which contains information on whether the joint location for that point has been determined (and if so what it is), pointers to the next and last records (corresponding to the previous and last intermediate points, respectively), and a pointer to the list of all of the obstacle avoiding path segments that end at that

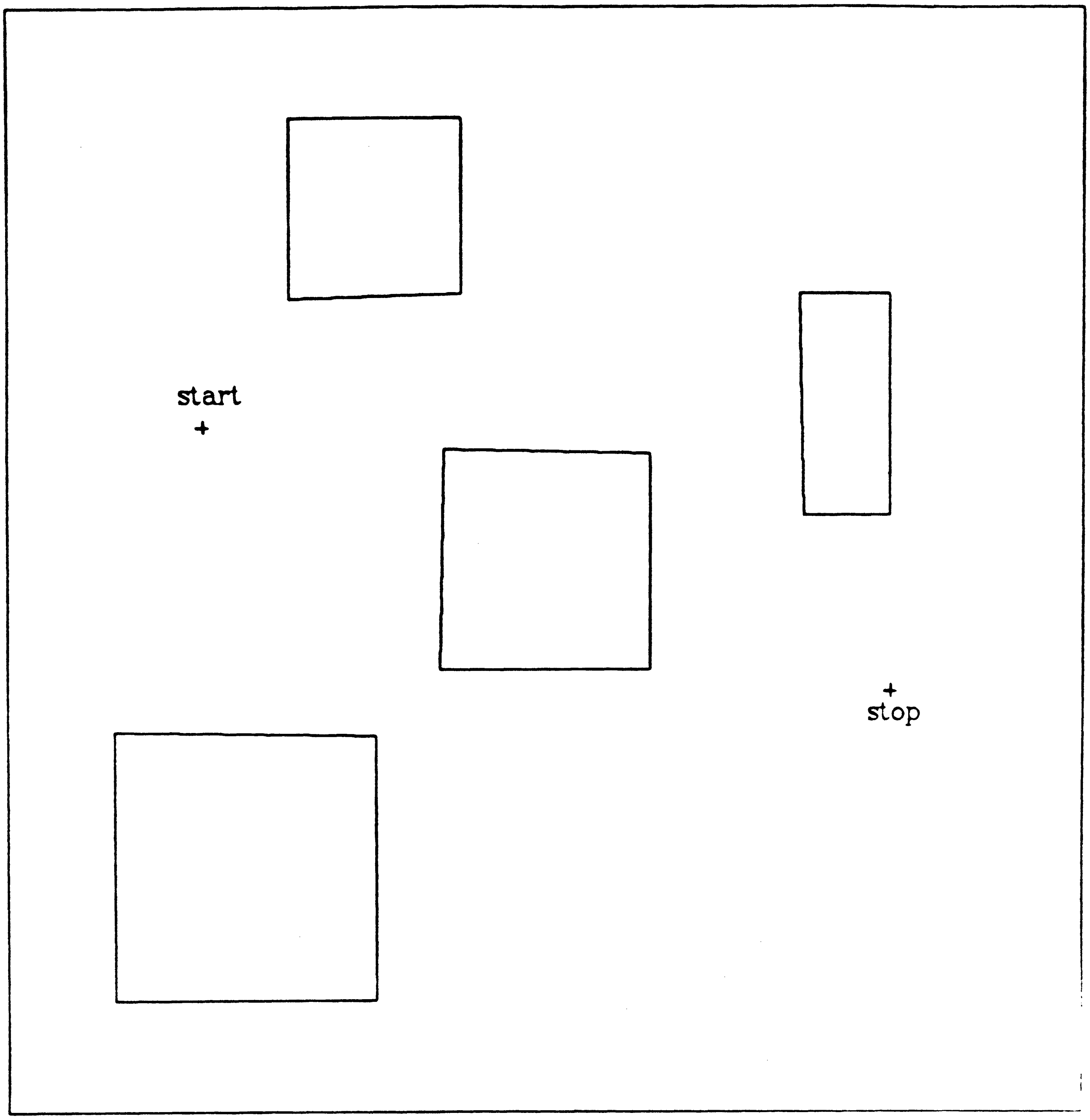

Figure 1: A room containing convex polygon obstacles and a pseudo obstacle.

point. The top row in Figure 2 corresponds to this linked list of records, each record corresponding to an intermediate point. We then create two pointers, P_{to} and P_{from}, and use the following algorithm to generate all of the path segments. (In what follows, to "increment" a pointer P means to set P to the *next* pointer in the record that P currently points to.)

Generate All Path Segments

1. Set both P_{to} and P_{from} to point to the record corresponding to the initial point $P_{initial}$ (or point number 0). This is the first record in the linked chain of records associated with intermediate points and will henceforth be called the *initial record.*
2. If P_{to} is equal to NULL, then stop.
3. If P_{to} is equal to P_{from}, then increment P_{to} and set P_{from} to the initial record and go to step 2.
4. Generate a path segment from the point in the record pointed to by P_{from} to the point in the record pointed to by P_{to}.
5. If the path segment intersects any obstacle or is outside of the manipulators' workspace, P_{from} points to the last record before the record pointed to by P_{to}, and the record P_{to} points to has no obstacle avoiding path segments connecting its intermediate point with any other intermediate point, then insert a new record with a new intermediate point (using *Generate New Intermediate Point*) between the records pointed to by P_{to} and P_{from}. Set P_{to} to point to this new record, and P_{from} to point to the initial record. Go to step 4.
6. If the path segment intersects any obstacles or is outside of the manipulator's workspace, increment P_{from} and go to step 3.
7. Create a record with fields identifying the origin ("from") point and terminal ("to") point and a pointer, *crv*, pointing to the path segment generated in step 4. Link this record at the end of P_{to}'s *curves* list (see Figure 2) and increment P_{from}. Go to step 3.

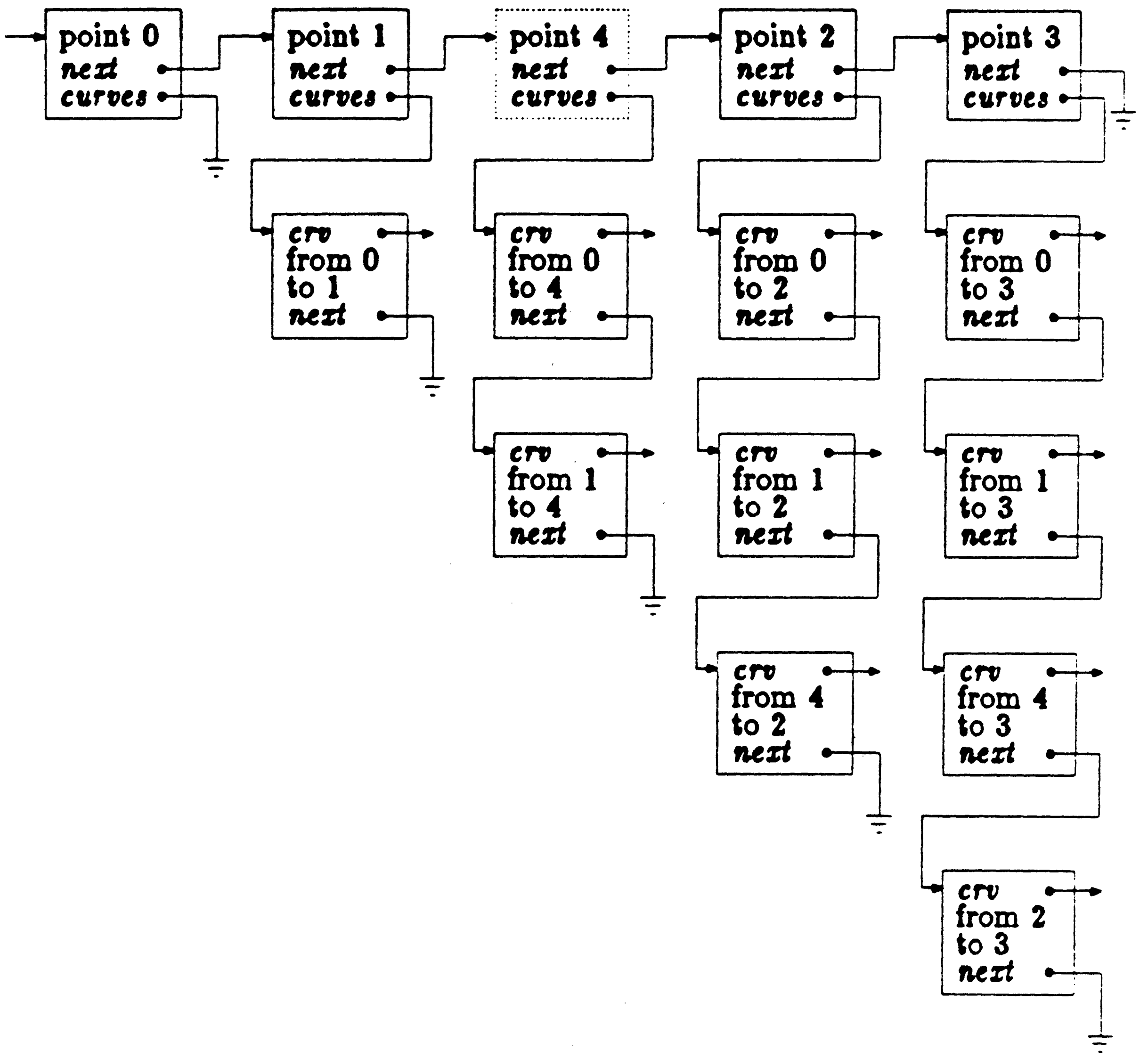

Figure 2: Data structure after Generate All Path Segments algorithm. Note the inclusion of a new intermediate point (point 4 shown in the dashed box).

This algorithm requires $O(N^2)$ time, where N is the number of points along the path (including $P_{initial}$ and P_{final}).

It should be noted that since many manipulators have more than one set of joint positions for each point in Cartesian space, it is important to assign a unique joint position to each intermediate point. This will enable us to join all combinations of path segments that originate or terminate at each intermediate point since each intermediate point will correspond to only one point in joint space. (If all possible joint positions were used for each intermediate point, and a path segment generated for all of the combinations, the number of possible paths would grow very rapidly.) In the preceding algorithm, the joint position that corresponds to each intermediate point is the joint position corresponding to the first acceptable path segment ending at that intermediate point[3].

Because of the way in which the intermediate points are selected, there *must* always exist a path composed of straight line segments connecting the initial and final positions. If the manipulator strays too far off this path (in following a geodesic) and intersects any obstacles, we will choose a new intermediate point along one of the the straight line segments making up this path. This is the basis for the method used for generating a new intermediate point.

Generate New Intermediate Point

The geodesic path segments are parameterized by their arc length (s) in inertia space and are sampled at equal intervals of s. Hence if $i_{intersect}$ is the index into the sampled path segment where the segment first intersects an obstacle, and there are N_{curve} sample points in the segment, then $t = i_{intersect}/N_{curve}$ represents the fraction of the total segment length covered before the collision. We select the next intermediate point at the same fractional distance along the straight line connecting the intermediate points. By selecting each new intermediate point this way, eventually an intermediate point will be chosen so that the geodesic path will return to a point along the line segment before intersecting any obstacle. The new

[3]Whenever a path segment is generated between two Cartesian points, the joint position of the origin point will be known. If more than one joint position corresponds to the terminal point, we choose the closest to the initial point in joint space.

intermediate points are computed as:

$$
\begin{aligned}
x_{new} &= x_{from} + (x_{to} - x_{from}) * t \\
y_{new} &= y_{from} + (y_{to} - y_{from}) * t \\
z_{new} &= z_{from} + (z_{to} - z_{from}) * t
\end{aligned}
$$

where

$(x_{from}, y_{from}, z_{from})$	=	the origin point of the line segment.
(x_{to}, y_{to}, z_{to})	=	the terminal point of the line segment,
$(x_{new}, y_{new}, z_{new})$	=	the new intermediate point location.

Note that since the fraction t will always be less than one, each new intermediate point (if more than one is needed) gets closer to the starting point along the straight line formed by the original end points of the segment. Since a straight line between intermediate points will not intersect any obstacles, and since the geodesic (or any other path type) will eventually shrink to an essentially straight line segment, we will eventually be able to generate a collision-free path this way. In addition, any $t \in (0,1)$ will work, even if it remains fixed. The new intermediate point is labeled with a number $p+1$, where the highest previous intermediate point label was p. Figure 3 shows graphically the selection of a new intermediate point.

This method of intermediate point selection does not always lead to the "best" choice of intermediate points, since more points than necessary are sometimes created. However, since our final path is not required to pass through all of the intermediate points, this effect will generally be diminished.

At the completion of this stage all of the path segments have been constructed and there exists at least one path from the initial point to the final point. We now have a structure like that pictured in Figure 2. Note that the structure in Figure 2 includes a new intermediate point (point 4) which is shown in the dashed box. In addition, none of the path segments in the figure intersect any of the obstacles. In many cases a number of path segments would intersect one or more of the obstacles and hence the pointers to them would be set to NULL. Now we must construct a tree which will allow us to determine all of the paths from the initial point,

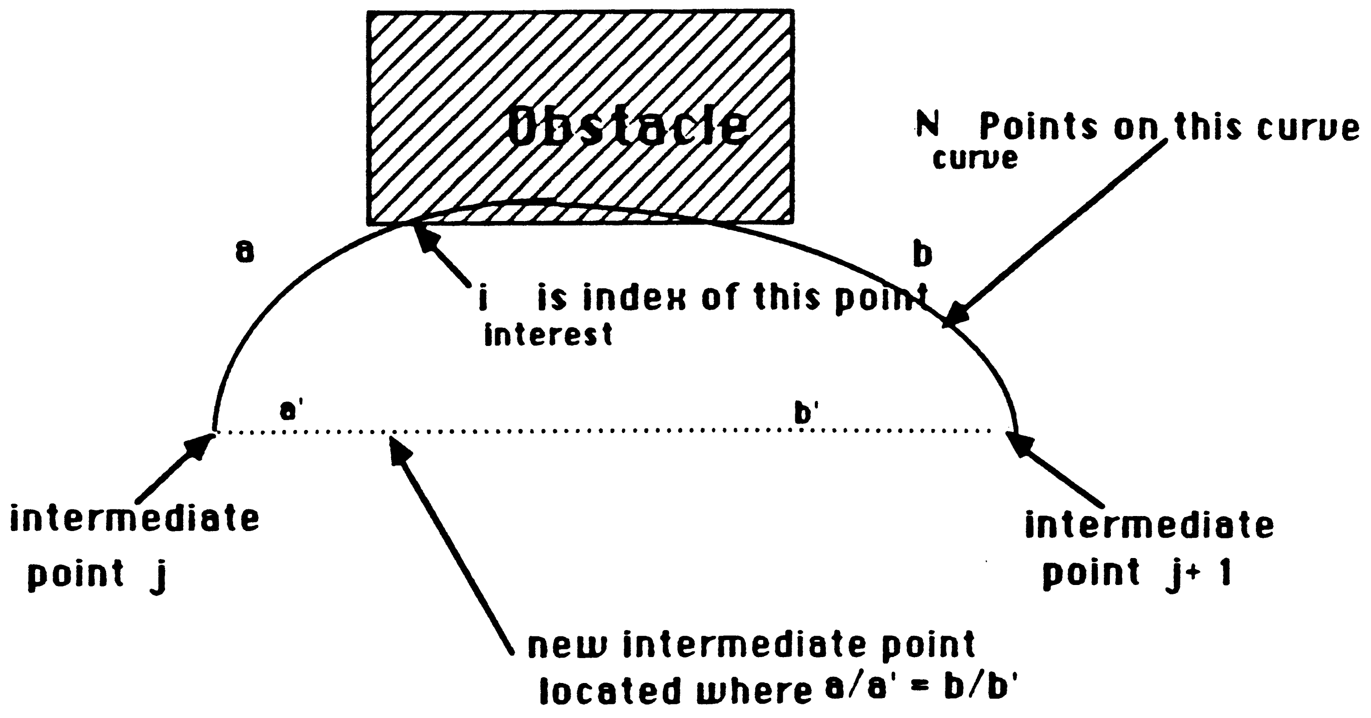
Obstacle
N_{curve} Points on this curve
a
b
$i_{interest}$ is index of this point
a'
b'
intermediate point j
intermediate point j+ 1
new intermediate point located where a/a' = b/b'

Figure 3: Graphical illustration of the selection of a new intermediate point.

$P_{initial}$, to the final point, P_{final}, which can be constructed from the path segments.

3.3 Construction of Solution Path Tree

The data structure used in constructing all of the obstacle avoiding path segments does not lend itself well to the determination of all the possible paths from the initial to the final point. We would like a data structure like that depicted in Figure 4, which contains records with the following pointers.

- *crv*: points to the path segment records generated by the *Generate All Path Segments* algorithm. Since we do not want to waste time copying segments already generated, we just use pointers to the paths already generated.
- *next path*: points to the record corresponding to the next path segment which originates at the current segment's endpoint. This next segment is one of the possible segments that could be joined with the current segment to produce a path from the initial to final positions.
- *same origin*: points to the record corresponding to the next path segment with the same origin point as the current segment.

In order to generate this type of structure, the following recursive algorithm is used.

Build Path Tree

1. Create a record and set the pointer P_{tree} to point to this record. The first path must originate at the point 0, which corresponds to the manipulator's starting position. Search through all path segments until the first segment with point 0 as its starting point is found. Set the pointer *crv* to point to this path segment.
2. Initialize pointers $O_{ptstart}$ and $P_{ptstart}$ to the first record of the intermediate point list.

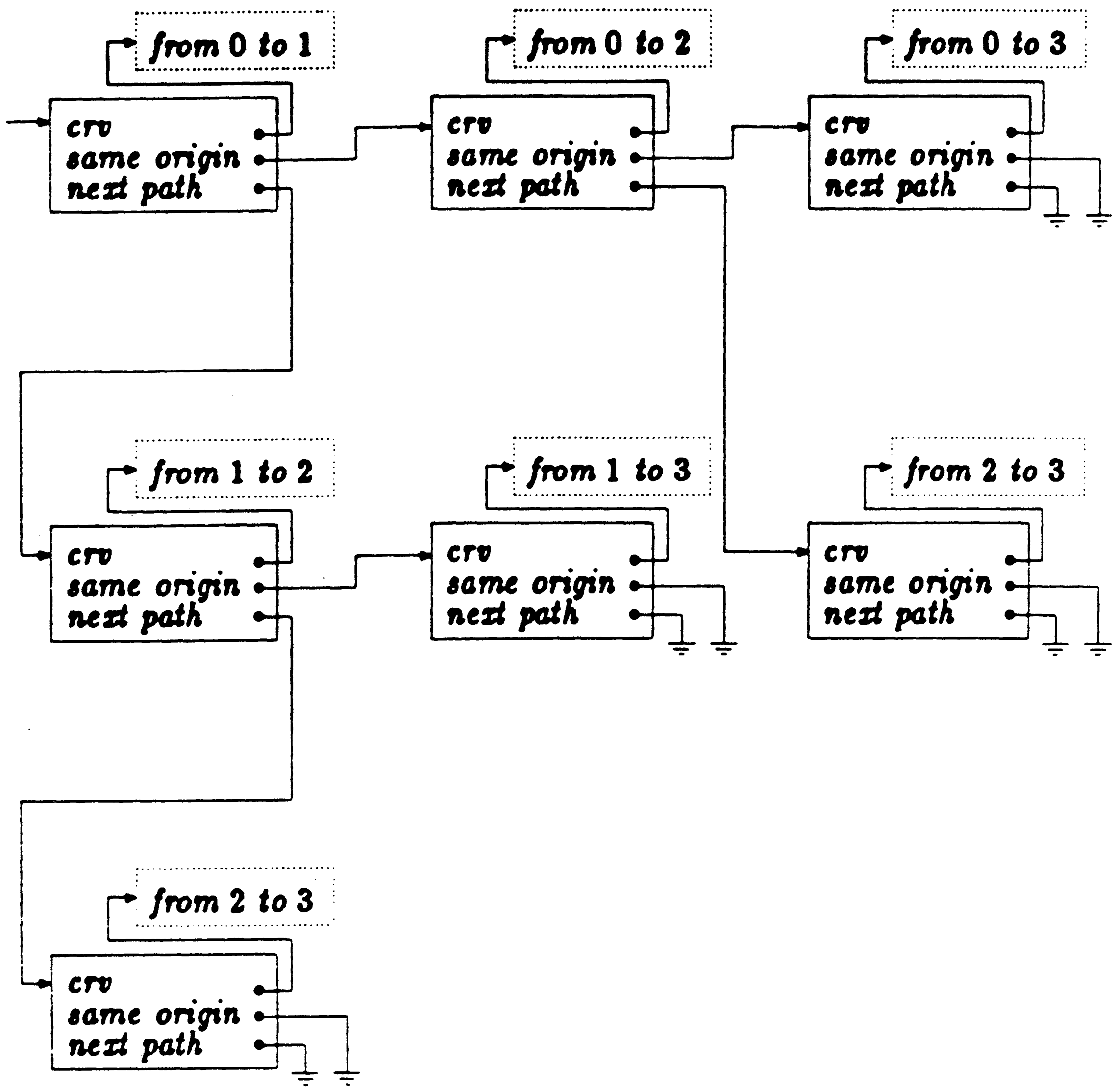

Figure 4: Data structure after Build Path Tree algorithm.

3. (The main procedure with three arguments, P_{tree}, $O_{ptstart}$, $P_{ptstart}$, begins here.)

   Determine the origin point of the path segment in the record pointed to by P_{tree}, and search the intermediate point list starting at the point pointed to by $O_{ptstart}$ for the next path segment with the same origin. Update $O_{ptstart}$ to point to the record corresponding to this segment. If no such segment is found, set $O_{ptstart}$ to NULL.

4. If $O_{ptstart}$ is equal to NULL, set $P_{neworigin}$ to NULL and go to step 5. Otherwise, create a new record and set the *crv* of this record to point to the segment found in step 3. Set $P_{neworigin}$ to point to this record and set the *next origin* of the record pointed to by P_{tree} to point to this new record.

5. Determine the end point of the segment in the record pointed to by P_{tree}, and search the intermediate point list starting at the point pointed to by $P_{ptstart}$ for the next segment with its origin equal to that end point. Update $P_{ptstart}$ to point to the record corresponding to this segment. If no such segment is found, set $P_{ptstart}$ to NULL.

6. If $P_{ptstart}$ is equal to NULL, set $P_{newpath}$ to NULL and go to step 7. Otherwise, create a new record and set the *crv* of this record to the segment found in step 5. Set $P_{newpath}$ to point to this new record. Set the *next path* of the record pointed to by P_{tree} to point to this new record.

7. If $P_{neworigin}$ is equal to NULL go to step 8, else go to step 3 with arguments $P_{neworigin}$, $O_{ptstart}$, $O_{ptstart}$.

8. If $P_{newpath}$ is equal to NULL then done, else go to step 3 with arguments $P_{newpath}$, $P_{ptstart}$, $P_{ptstart}$.

This algorithm requires $O(2^N)$ time, where N is the number of points along the path (including $P_{initial}$ and P_{final}).

At this point we have a structure like the example shown in Figure 4. (No new intermediate points were used in this figure.)

Note that we have determined em all possible loop-free paths. From this structure, determining all of the path segments that make up a path is straightforward.

3.4 Connecting the Path Segments into a Continuous Path

In order to determine the minimum traversal time along a path, the phase plane algorithm presented by Shin and McKay [10] is used, since this method determines the true minimum time trajectory for a parameterized path utilizing the full robot dynamics. Since the phase plane method used to derive the minimum time trajectory requires continuous first and second derivatives, the parameterized path must have continuous first and second derivatives. Fifth order polynomials (quintics) are used to join path segments (in joint space) so that the position, first, and second derivatives at the sample points where the two path segments are joined match. Because our paths are parameterized by arc length in inertia space, the quintics should also be parameterized by their arc length in inertia space. However, due to the highly nonlinear form of the inertia matrix, this arc length cannot be determined *a priori*. The following simple algorithm is used to determine the quintic curves as a function of their arc length.

1. Make an initial estimate of the arc length in inertia space by assuming the two sample points are to be joined by a joint interpolated path. For each joint i we compute $d^i = q^i_{final} - q^i_{initial}$. Now break this distance up into $N - 1$ equal parts, $\delta^i = d^i/(N-1)$, $N > 1$. At each of these $N-1$ points, estimate the arc length from the last point using $ds \approx \sqrt{J_{ij}dq^i dq^j}$ and sum these to estimate the total arc length, S_{max}.

2. Using this estimate of S_{max}, compute the quintics which have the same position, first, and second derivatives as the (joint space) path segments joined at $s = 0$ (the sample point along the first path) and at $s = S_{max}$ (the sample point along the second path).

3. Divide S_{max} into $N - 1$ equal parts, and compute a new estimate of S_{max}.

4. If the change in estimates of S_{max} is within a specified tolerance then stop; otherwise go to step 2 with the new estimate of S_{max}.

With the above method for joining two path segments to make a continuous path, the remaining problem is to determine which sample point on the first path segment should be joined with which sample point on the second segment. If the functions used to join intermediate points are nearly optimal (locally), then when connecting path segments together, we would like the original segments to retain their basic shapes[4] (straight lines or geodesics in our case). Otherwise, we may be substituting quintic curves for one that was nearly optimal. However, the resulting path should be reasonably smooth with no sharp corners. Let us define the *head curve* as the portion of the path which has already been generated, and the *tail curve* as the path segment we wish to join to the head curve. Then the following algorithm is used to determine the sample points used in joining two path segments.

Choose Joining Points

Given:

N_{last}	=	the number of points in the last segment of the head curve
N_{tail}	=	the number of points in the tail curve
α	=	the maximum percent of each curve's points to be used
$N_{minimum}$	=	the minimum number of points from each curve to use
P_{head}	=	the number of points in the head curve to try
P_{tail}	=	the number of points in the tail curve to try.

1. Compute $Start_{head}$ as the initial index into the head curve to begin the search, and set $Stop_{head}$ to the last index into the head curve. Set $Start_{tail}$ to 0 and compute $Stop_{tail}$. The calculations for $Start_{head}$ and $Stop_{tail}$ are based on N_{last}, α, $N_{minimum}$, and N_{tail}.

[4]However, if one of the segments to be joined has fewer than a prescribed amount of sample points, it may be essentially eliminated when a quintic curve is joined to the beginning and end of the segment.

2. Compute

$$\begin{aligned}\delta_{head} &= (Stop_{head} - Start_{head})/P_{head}\\ \delta_{tail} &= (Stop_{tail} - Start_{tail})/P_{tail}.\end{aligned}$$

3. Join all sample points in the head curve which are indexed by

$$i_{head} = Start_{head} + k\delta_{head}, \quad k = 0, \ldots, (P_{head} - 1),$$

with all sample points in the tail curve defined which are indexed by

$$i_{tail} = Start_{tail} + j\delta_{tail}, \quad j = 1, \ldots, P_{tail}$$

with quintic curves.

4. For each joint i, compute t_i, where t_i is the average value (over all of the sample points on the quintics) of the absolute value of the curvature in inertia space. The curvature in inertia space is given as [14]

$$\frac{\delta p^i}{\delta s} = \frac{dp^i}{ds} + \left\{ \begin{matrix} i \\ jk \end{matrix} \right\} p^j \frac{dq^k}{ds}$$

where

q^i	=	the position of joint i,
s	=	the arc length in inertia space,
p^i	=	dq^i/ds,
$\left\{ \begin{smallmatrix} i \\ jk \end{smallmatrix} \right\}$	=	the Christoffel symbol of the second kind.

5. The quintics which minimizes $T = \sum_i t_i$, while avoiding all of the obstacles and remaining within the manipulators workspace, are used to join the head and tail curves together.

The number of sample points in the joining quintic polynomials was equal to the larger of a predefined minimum or the number of sample points being replaced by the quintics. The quintic samples were equally spaced in the parameter s.

If all of the quintic curves generated using the above procedure either intersect an obstacle or are outside of the manipulator's workspace, then an alternative procedure must be used. Starting with the lowest index tested in the head curve and the highest index tested in the tail curve, generate the joining quintic curves. Determine the index in the joining curves which corresponds to the first sample point intersecting an obstacle (or going outside of the manipulator's workspace). Determine which curve this sample point corresponds to (either the head or the tail curve). If the sample point corresponds to the head curve, then increment the index in the head curve used to start the joining quintics, otherwise decrement the index into the tail curve used. Now, go back and generate new joining quintics using the new indices into the two curves. Eventually we must be able to find two indices into the head and tail curve which correspond to sample points which can be joined without the joining quintics intersecting any obstacles, since the head and tail curves do not intersect any obstacles.

4 Simulation Results

A simulation of the proposed method for the first three joints of the Stanford manipulator was written in C and run on a VAX[5] 11/780 under the UNIX[6] 4.2 operating system. The results of two of these simulations are presented in Tables 1 - 4.

The common parameters for both simulations are $P_{head} = 8$, $P_{tail} = 8$, and $N_{minimum} = 20$. The only difference between simulation runs was that in the first $\alpha = 0.25$ while in the second $\alpha = 0.10$. The parameter α indicates the percentage of each path segment can be replaced with a quintic polynomial when joining segments. $N_{minimum}$ represents the minimum number of sample points from each segment involved in the joining process (unless the segment had fewer than $N_{minimum}$ points). P_{head} and P_{tail} indicate the number of sample points from the head curve and that from the tail curve to be joined with quintic polynomials. With both P_{head} and P_{tail} set to 8, there will be a total of 64 quintic joining paths to choose from. It

[5]VAX is a trademark of Digital Equipment Corporation.

[6]UNIX is a trademark of AT & T Bell Laboratories.

should be noted that if 25 percent of each path segment is replaced with a quintic polynomial, then at most 50 percent of the total path can be composed of quintic polynomials.

For the room and obstacles presented in Figure 1, there are two possible path directions. The first direction is above the pseudo obstacle using five intermediate points (labeled with numbers in squares), while the second direction is below the pseudo obstacle using three intermediate points (labeled with numbers in circles). The points used for the "top" and "bottom" paths are shown in Figure 5.

Table 1 contains the results for the "top" paths for $\alpha = 0.25$, and is sorted by increasing traversal time. The left column indicates the points used (point 0 is the initial point and point 6 is the final point), the middle column indicates the traversal time of path made up of straight line segments, and the final column represents the traversal time for paths made up of geodesic segments. For this "top" path there were 16 paths made up of straight line segments and 12 paths made up of geodesic segments. The quickest times for all "top" paths were paths constructed of straight line segments. Note, however, that in any case where both the straight line and geodesic path segments connected the same intermediate points, the path made up of geodesic segments could be traversed in less time than the path made up of straight line segments.

Table 2 presents the results of the "bottom" paths for $\alpha = 0.25$, in increasing order of traversal time. Note that there is no overlap between the geodesic and straight line paths. This is because an intermediate point, point 5, was needed for the geodesic paths to get from the intermediate point 2 to intermediate point 3, while no such intermediate point was needed for the straight line segments. Nearly all of the "bottom" paths are quicker than the fastest of the "top" paths.

Tables 3 and 4 present similar results of the simulation runs with $\alpha = 0.10$. Table 3 presents the results for the "top" paths, while Table 4 presents the results for the "bottom" paths. Note that in all cases the traversal times have increased, and in a few cases the ordering has been changed slightly.

Tables 5 and 6 present additional results of the simulation for the the "top" paths for both geodesic and straight line path segments (the ordering of the paths is in the order the simulation generated them, unlike the

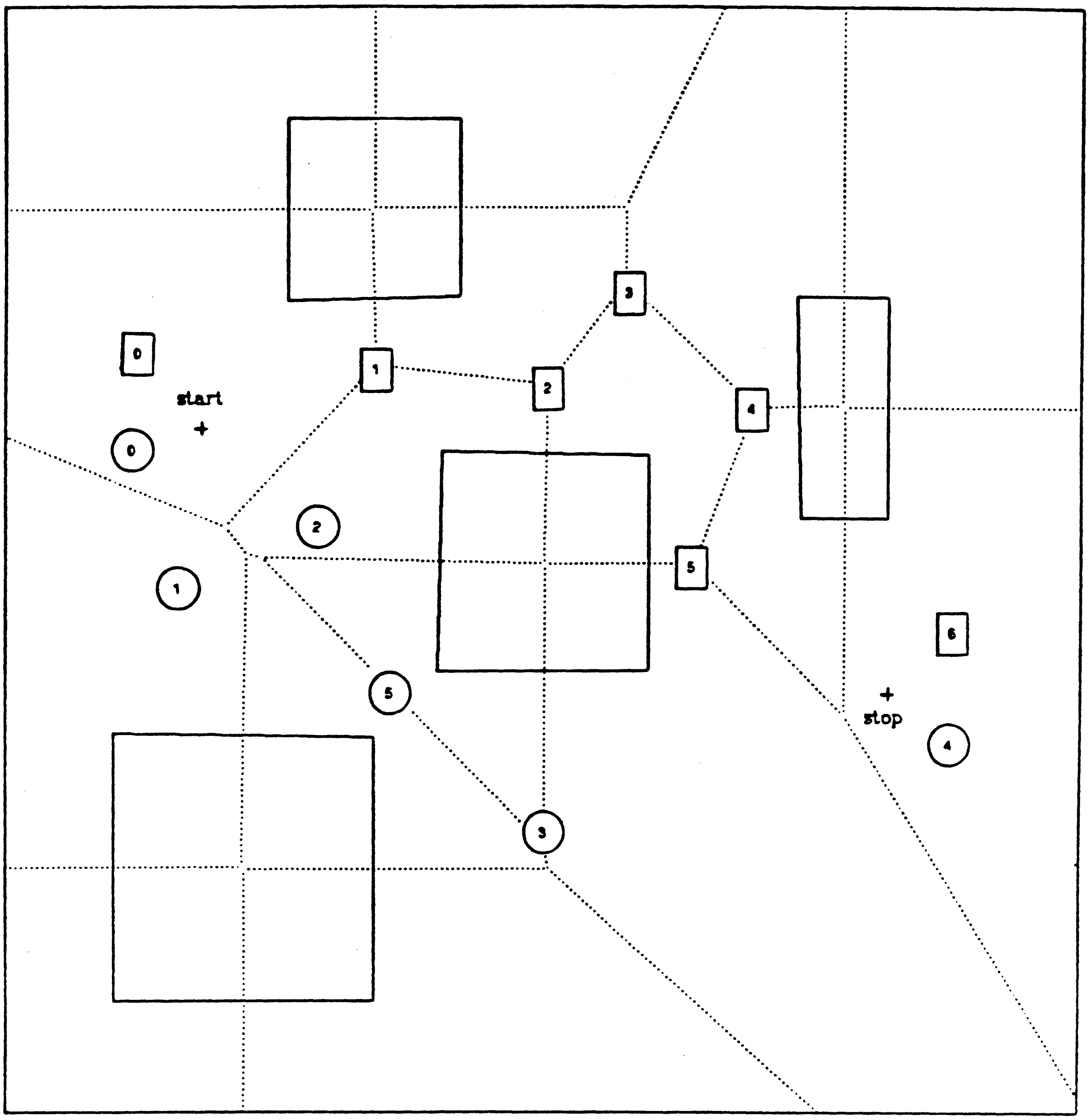

Figure 5: Points used for "top" and "bottom" paths.

Path	Line Segments	Geodesic Segments
$0 \rightarrow 3 \rightarrow 6$	2.588	—
$0 \rightarrow 1 \rightarrow 3 \rightarrow 6$	2.655	—
$0 \rightarrow 2 \rightarrow 5 \rightarrow 6$	—	3.074
$0 \rightarrow 1 \rightarrow 5 \rightarrow 6$	—	3.078
$0 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 6$	—	3.081
$0 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.316	3.144
$0 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.652	3.252
$0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.461	3.272
$0 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.279	—
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6$	3.290	—
$0 \rightarrow 2 \rightarrow 3 \rightarrow 6$	3.304	—
$0 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.624	3.339
$0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.565	3.354
$0 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.513	—
$0 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.910	3.717
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.896	3.721
$0 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	4.047	3.857
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	4.032	3.861
$0 \rightarrow 4 \rightarrow 5 \rightarrow 6$	4.450	—

Table 1: Simulation results for the "top" paths for $\alpha = 0.25$.

Path	Line Segments	Geodesic Segments
$0 \rightarrow 2 \rightarrow 5 \rightarrow 4$	—	2.163
$0 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 4$	—	2.198
$0 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 4$	—	2.200
$0 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 4$	—	2.235
$0 \rightarrow 1 \rightarrow 3 \rightarrow 4$	2.460	—
$0 \rightarrow 2 \rightarrow 3 \rightarrow 4$	2.483	—
$0 \rightarrow 3 \rightarrow 4$	2.580	—
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$	2.596	—

Table 2: Simulation results for the "bottom" paths for $\alpha = 0.25$.

Path	Line Segments	Geodesic Segments
$0 \to 3 \to 6$	2.702	—
$0 \to 1 \to 3 \to 6$	2.727	—
$0 \to 2 \to 5 \to 6$	—	3.190
$0 \to 1 \to 5 \to 6$	—	3.206
$0 \to 1 \to 2 \to 5 \to 6$	—	3.206
$0 \to 1 \to 3 \to 5 \to 6$	3.420	3.309
$0 \to 2 \to 3 \to 6$	3.365	—
$0 \to 1 \to 2 \to 3 \to 6$	3.374	—
$0 \to 1 \to 4 \to 5 \to 6$	3.742	3.407
$0 \to 3 \to 5 \to 6$	3.409	—
$0 \to 1 \to 3 \to 4 \to 5 \to 6$	3.576	3.422
$0 \to 2 \to 4 \to 5 \to 6$	3.758	3.502
$0 \to 1 \to 2 \to 4 \to 5 \to 6$	3.719	3.502
$0 \to 3 \to 4 \to 5 \to 6$	3.610	—
$0 \to 2 \to 3 \to 5 \to 6$	4.032	3.961
$0 \to 1 \to 2 \to 3 \to 5 \to 6$	4.041	3.957
$0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 6$	4.194	4.072
$0 \to 2 \to 3 \to 4 \to 5 \to 6$	4.184	4.075
$0 \to 4 \to 5 \to 6$	4.230	—

Table 3: Simulation results for the "top" paths for $\alpha = 0.10$.

Path	Line Segments	Geodesic Segments
$0 \to 2 \to 5 \to 4$	—	2.217
$0 \to 1 \to 2 \to 5 \to 4$	—	2.276
$0 \to 2 \to 5 \to 3 \to 4$	—	2.316
$0 \to 1 \to 2 \to 5 \to 3 \to 4$	—	2.376
$0 \to 1 \to 3 \to 4$	2.563	—
$0 \to 2 \to 3 \to 4$	2.587	—
$0 \to 3 \to 4$	2.705	—
$0 \to 1 \to 2 \to 3 \to 4$	2.733	—

Table 4: Simulation results for the "bottom" paths for $\alpha = 0.10$.

ordering in Tables 1 - 4). In these tables we vary α from 0.10 to 0.60 while the other parameters remain as they were in the previous simulations. As α increases, more of the original curve segments are replaced by quintics. These tables show a general trend of smaller traversal times as α increases, though this is not always true. For example, as Table 6 shows, the minimum traversal time for the path connecting the intermediate points 0,4,5, and 6 made up of line segments initially increases with increasing α and then decreases.

This behavior can be explained as follows. We are joining the joint positions with quintics in joint space (not in Cartesian space). In addition, only a small number of possible joining points are tested, and we are choosing the joining quintics which minimize the absolute value of curvature in inertia space. Finally, straight line paths in Cartesian space do not correspond to straight line paths in joint space, and as we connect the line segments to form the Cartesian path the corresponding paths in joint space (one path for each joint) can have many inflection points. As we allow more of the paths to be quintics (α increasing), the quintics which minimize our cost function may jump over some of the local inflections in the joint curves. This behavior can cause the increase in traversal time.

The best improvement in traversal time between the paths composed of geodesic segments and those composed of straight line segments occurs when both paths connect the intermediate points 0,1,3,4,5 and 6. In this case, when $\alpha = 0.60$, the path of geodesic segments is approximately 15 percent faster. Figures 6 through 9 contain two dimensional projections[7] of the paths generated by the simulation for the situation presented in Figure 1. Figures 6 and 7 show the paths composed of straight line segments and geodesic segments connecting intermediate points 0,1,3,4,5 and 6 for $\alpha = 0.20$, while Figures 8 and 9 show the same paths for $\alpha = 0.60$.

These simulation results are typical of other cases we have examined. In nearly all cases the matching quintics joined the two farthest sample points allowed. In each case where there was a path made up of geodesic segments and a path made up of line segments, the path composed of geodesic segments could be traversed faster than the corresponding path composed of line segments. In those cases where the fastest time was for

[7]In these figures we are looking down on the workspace and robot.

Path	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	4.072	3.930	3.794	3.666	3.537	3.436
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.957	3.797	3.643	3.490	3.341	3.245
$0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.502	3.402	3.308	3.222	3.139	3.077
$0 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 6$	3.206	3.118	3.046	2.993	2.943	2.888
$0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.422	3.316	3.231	3.157	3.089	3.039
$0 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.309	3.193	3.096	3.002	2.917	2.853
$0 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.407	3.295	3.213	3.148	3.092	3.042
$0 \rightarrow 1 \rightarrow 5 \rightarrow 6$	3.206	3.110	3.048	2.989	2.934	2.876
$0 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	4.075	3.926	3.789	3.658	3.529	3.424
$0 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.961	3.793	3.638	3.483	3.333	3.229
$0 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.502	3.393	3.288	3.198	3.120	3.062
$0 \rightarrow 2 \rightarrow 5 \rightarrow 6$	3.190	3.101	3.046	2.993	2.942	2.890

Table 5: Simulation results for the "top" paths for geodesics.

Path	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	4.194	4.078	3.983	4.038	3.980	3.912
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$	4.041	3.940	3.852	3.767	3.691	3.583
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6$	3.374	3.313	3.266	3.218	3.175	3.108
$0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.719	3.608	3.521	3.594	3.560	3.534
$0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.576	3.494	3.426	3.510	3.492	3.485
$0 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.420	3.345	3.279	3.230	3.218	3.207
$0 \rightarrow 1 \rightarrow 3 \rightarrow 6$	2.727	2.673	2.623	2.588	2.573	2.573
$0 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.742	3.671	3.632	3.745	3.722	3.356
$0 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	4.184	4.085	4.000	4.067	4.022	3.923
$0 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$	4.032	3.948	3.869	3.796	3.732	3.538
$0 \rightarrow 2 \rightarrow 3 \rightarrow 6$	3.365	3.321	3.283	3.247	3.216	3.030
$0 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.758	3.664	3.581	3.656	3.622	3.595
$0 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	3.610	3.541	3.480	3.556	3.503	3.489
$0 \rightarrow 3 \rightarrow 5 \rightarrow 6$	3.409	3.325	3.253	3.226	3.227	3.212
$0 \rightarrow 3 \rightarrow 6$	2.702	2.616	2.575	2.574	2.571	2.564
$0 \rightarrow 4 \rightarrow 5 \rightarrow 6$	4.230	4.370	4.511	4.771	4.165	3.578

Table 6: Simulation results for the "top" paths for line segments.

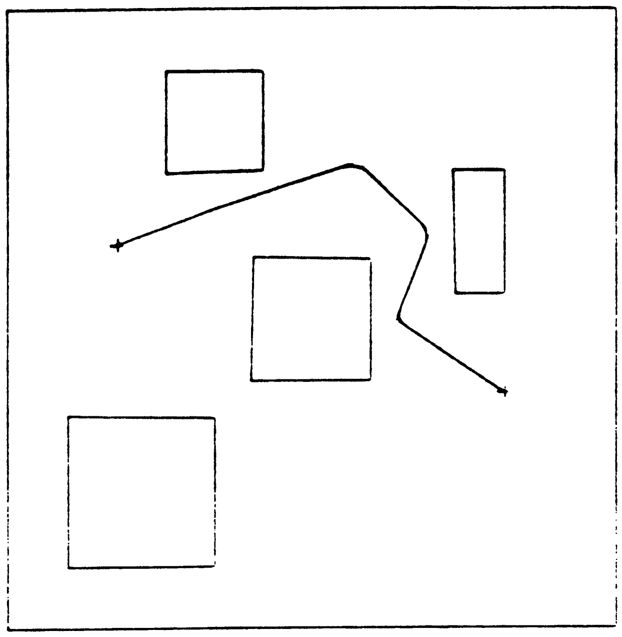

Figure 6: Path composed of line segments connecting intermediate points 0,1,3,4.5, and 6. ($\alpha = 0.20$, traversal time = 3.494 seconds.)

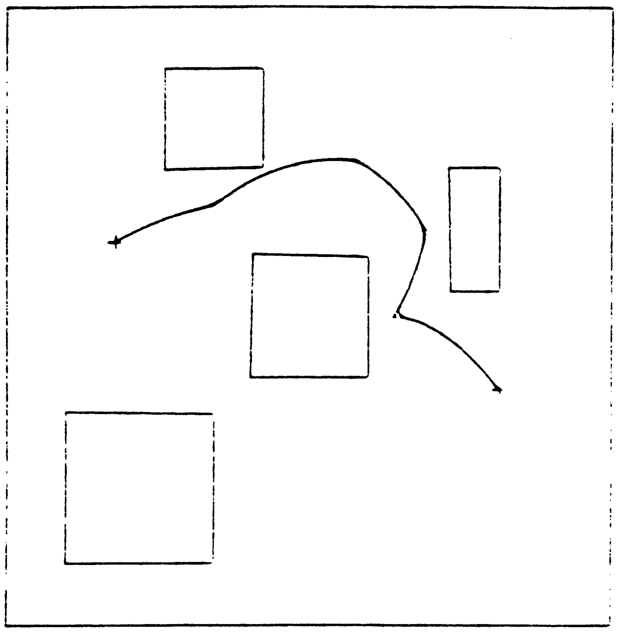

Figure 7: Path composed of geodesic segments connecting intermediate points 0,1,3,5 and 6. ($\alpha = 0.20$, traversal time = 3.316 seconds.)

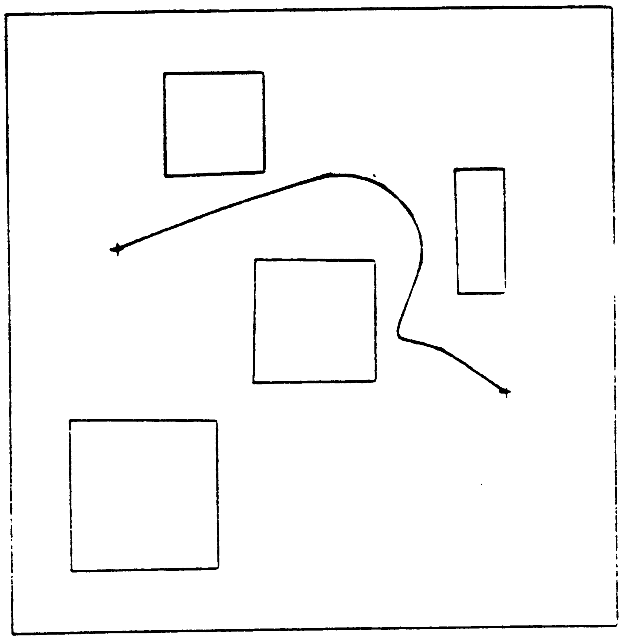

Figure 8: Path composed of line segments connecting intermediate points 0,1,3,4,5, and 6. ($\alpha = 0.60$, traversal time = 3.485 seconds.)

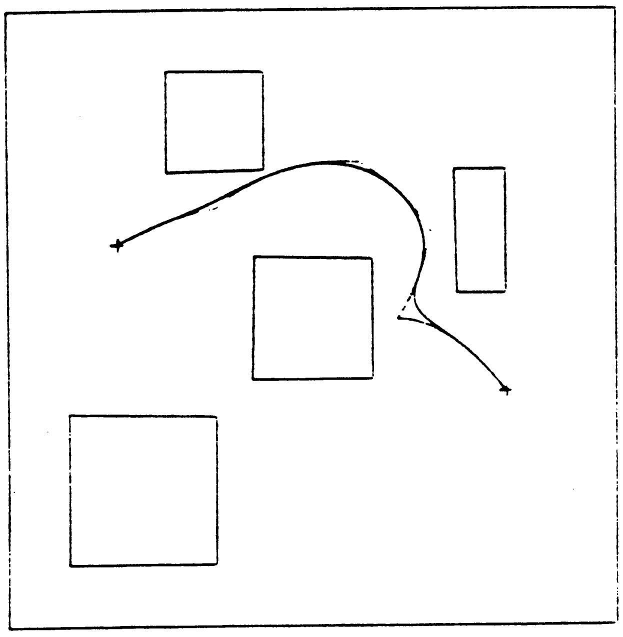

Figure 9: Path composed of geodesic segments connecting intermediate points 0,1,3,4,5, and 6. ($\alpha = 0.60$, traversal time = 3.039 seconds.)

a path composed of straight line segments, there was no path composed of geodesic segments connecting the same intermediate points.

5 Conclusions

We have developed a method of determining a continuous path for a manipulator's tip from an initial position to a final position in a room filled with convex polygon obstacles. Suitable intermediate points were selected initially using a Voronoi diagram made up of the corner points of the obstacles. All possible obstacle avoiding path segments connecting each intermediate point with each of the other intermediate points using a given path type were generated. All of the feasible paths constructed from these path segments from the initial to final positions were then constructed. Path segments were joined with the quintic polynomials which produced the minimum average absolute value of the curvature in inertia space. The minimum time required to traverse each trajectory generated was then computed.

Computer simulations for the first three joints of the Stanford manipulator indicated that in cases where both straight line and geodesic segments passed through the same intermediate points, the path made from geodesics could be traversed in less time. However, the path generated with the overall smallest traversal time could be composed of straight line segments only.

When joining two path segments, not all of the points along each segment are tested. However, our experience has been that in the majority of cases the joining path is that which joins the sampled points as far apart as we allow (as determined by the maximum allowed percentage of each path segment discarded).

As seen earlier, our method does present advantages over previous methods. We do not force the path to connect all of the intermediate points if the resulting path leads to a trajectory which can be traversed in less time. In addition, in using a near minimum time path for each segment, our solutions may offer a good suboptimal path. Our method also allows for the generation of new intermediate points whenever they are required.

References

[1] C. S. Lin, P. R. Chang, and J. Y. S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for mechanical manipulators," *IEEE Trans. Automat. Cont.*, vol. AC-28, no. 12, pp. 1066-1074, Dec. 1983.

[2] V. T. Rajan, "Minimum time trajectory planning," *Proc. IEEE Conf. Robotics and Automation* , March 1985, pp. 759-764.

[3] G. Sahar and J. M. Hollerbach, "Planning of minimum-time trajectories for robot arms," *Proc. IEEE Conf. Robotics and Automation* , March 1985, pp. 751-758.

[4] E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE Journal of Robotics and Automation*, vol. Ra-1, no. 1, pp. 21-30, March 1985.

[5] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control.* Cambridge, MA: M.I.T. Press, 1981.

[6] M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, eds., *Robot Motion: Planning and Control.* Cambridge, MA: M.I.T. Press, 1982.

[7] K. G. Shin and N. D. McKay, "Selection of near-minimum time geometric paths for robotic manipulators," *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 6, pp. 501-511, June 1986.

[8] B. K. Kim and K. G. Shin, "Minimum-time path planning for robot arms and their dynamics," *IEEE Trans. Sys., Man, and Cybernetics*, vol. SMC-15, no. 2, March 1985.

[9] J. Y. S. Luh and C. S. Lin, "Optimum path planning for mechanical Manipulators," *Jour. of Dynamic Sys., Measurement, and Control*, vol. 102, pp. 142-151, June 1981.

[10] K. G. Shin and N. D. McKay, "Minimum-time control of a robotic manipulator with geometric path constraints," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 6, pp. 531-541, June 1985.

[11] C. O. Dúlaing and C. K. Yap, "The Voronoi method for motion-planning : I. The Case of a disk," *Journal of Algorithms*, vol. 6, pp. 104-111, 1985.

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

[13] J. Y. S. Luh and C. Campbell, "Minimum distance collision-free path planning for industrial robots with a prismatic joint," *IEEE Trans. Automat. Control*, vol AC-29, no. 8, pp. 675-680, August 1984.

[14] J. L. Synge and A. Schild, *Tensor Calculus*, New York: Dover, 1978.