GEOMETRICAL MOTION PLANNING OF MANIPULATORS USING A CONTINUOUS CURVATURE MODEL

BENJAMIN J. KUIPERS & AKIRA HAYASHI

SEPTEMBER 1989   AI89-114
Geometrical Motion Planning of Manipulators
using a Continuous Curvature Model

Benjamin J. Kuipers
Akira Hayashi
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

October 3, 1989

\[ \text{1This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grant IRI-8602665 and by NASA grants NAG 2-507 and NAG 9-200. Akira Hayashi has been supported by Microelectronics and Computer Development Fellowship from the University of Texas at Austin.} \]
Abstract

We are trying to build an on-line motion planner for robot manipulators. It is intuitively obvious that the movement of a manipulator becomes more flexible as we increase the degree of freedom and thus make the manipulator system more redundant. But on the other hand, known algorithms for motion planning are intractable in terms of the degree of freedom, preventing us from designing a more flexible manipulator. Here we propose a new approach to the motion planning problem. We propose a planning system using a continuous curvature model which takes advantage of redundancy in motion planning without a combinatorial explosion. The continuous curvature model is an idealized manipulator which has an infinite number of rotational joints and is controlled by its curvature. After we get a solution for the continuous curvature model, we try to approximate it by an actual manipulator. By using the continuous curvature model, we can build a motion planning algorithm whose complexity is not related to the number of joints of an actual manipulator but to the complexity of its working environment, while taking advantage of its redundancy. The goal of our approach is to build an efficient and robust motion planner which could be applied to on-line control of a manipulator. In this paper, we introduce two early results of our approach. One is a design of a motion planning system using the continuous curvature model. The other is a simulation for testing some of the building blocks of the motion planning system. On the basis of these encouraging results, we claim that our approach has a significant potential and deserves further research.
5.2.6 Are There Such Manipulators? .............................................. 27

A Curvature Equation and its Numerical Solution 28
  A.1 Curvature Equation ........................................................... 28
  A.2 Numerical Solution of Curvature Equation ........................... 28
List of Figures

1 Scenario for Achieving Goal Position ........................................ 1
2 Solution Sequence of Our Approach ........................................... 3
3 Curvature Segment Representation and its Operators ...................... 5
4 Polar Coordinates .................................................................... 6
5 Curvature Segment Type ............................................................ 8
6 Decomposition of Segment ....................................................... 8
7 Approximation with a Multi-link Manipulator ............................... 8
8 Motion Planning System ............................................................ 9
9 Free Space Representation .......................................................... 10
10 Hypothesizing a Final Configuration ......................................... 11
11 Critical Points ......................................................................... 12
12 Plan Representation ................................................................... 12
13 Touch means New Constraint, not Failure ................................... 14
14 Open Space Problems for Decomposed Segments ......................... 15
15 Simulation Window .................................................................... 16
16 Successful Hill-Climbing ............................................................ 18
17 Local Maximum in Hill Climbing .............................................. 19
18 Five Initial Configurations ......................................................... 20
19 Hill Climbing with Good Initial Configuration ............................. 20
20 Example 1 as OSP1 ................................................................... 21
21 Example 1 as OSP2 ................................................................... 22
22 Example 1 as OSP3 ................................................................... 22
23 Example 1 as OSP4 ................................................................... 23
24 Example 2 as OSP4 ................................................................... 23
25 Example 3 as OSP4 ................................................................... 24
1 Introduction

We are interested in motion planning of a robot manipulator and are trying to efficiently solve the find path problem. The find path problem is that of finding a collision free path (a sequence of motions) for a robot in a known environment. In this paper we consider the problem for a manipulator in two dimensional space.

1.1 Use of Redundancy for Obstacle Avoidance

It is intuitively obvious that the movement of a manipulator becomes more flexible as we increase the degree of freedom and thus make the manipulator system more redundant. A redundant system allows us to pursue different goals at the same time, reaching a position while avoiding obstacles, for example. But on the other hand, known algorithms for motion planning are intractable in terms of the number of joints ([Schwartz and Sharir, 1983], [Lozano-Pérez, 1986],[Canny, 1988]), preventing us from designing and utilizing a more redundant manipulator. Although there have been some research on utilizing redundancy for obstacle avoidance ([Yoshikawa, 1984],[Kirčanski and Vukobratović, 1986]), their use of redundancy is not aggressive. In [Kirčanski and Vukobratović, 1986], they propose to utilize redundancy for obstacle avoidance when a desired manipulator end-effector trajectory is given. Thus their problem is not the find path problem. Here we propose a new approach to the problem in which redundancy is utilized more aggressively from the first stage of motion planning.

1.2 Swan’s Neck Scenario

An example may be helpful to explain our main ideas. Let’s think of a swan’s neck as a highly redundant manipulator and see Figure 1. When the swan’s neck is at 1, how can the swan reach the goal shown in the

![Figure 1: Scenario for Achieving Goal Position](image-url)
figure? Our scenario for his movement is like this. The swan somehow notices that he must first clear the narrow channel between the obstacle and the neck base. So he sets the subgoal at somewhere in the channel and try to reach the subgoal first. After he reaches the subgoal (at 4), he then tries to reach the goal, and this situation is where redundancy works. In order to reach the goal, he has to extend his neck further. In order not to collide with the obstacle, he has to keep the subgoal position. So he tries to attain the subgoal with a shorter length while the rest of the neck tries to reach the goal from the subgoal (at 5 and 6).

1.3 Our Approach

Notice that a flexible and continuous change of shape is assumed in the scenario. Accordingly, we propose a motion planning system using a continuous curvature model which can simulate the above scenario. The continuous curvature model is an idealized manipulator which has an infinite number of rotational joints. It has been developed to represent a highly redundant manipulator in a compact manner. Its motion is controlled by its curvature. Its configuration is represented by a series of curvature segments. The number of curvature segments is controlled by a decomposition technique to dynamically control the degree of redundancy.

By using the continuous curvature model, we can build a motion planning algorithm whose complexity is not related to the number of joints of an actual manipulator but to the complexity of its working environment, while taking advantage of its redundancy. At the initial stage of motion planning, the manipulator's work space is searched to find a set of subgoals to achieve the given task. Sequential subgoals will become steps of the plan, and competing subgoals such as reaching a position and avoiding obstacles can be resolved by dynamically introducing a redundancy using the decomposition technique. After introducing the redundancy, each subgoal can be pursued quite independently by the corresponding curvature segment, and a combinatorial explosion is avoided. Hill-climbing search in curvature space is conducted to achieve the subgoals. Thanks to the continuous curvature model, various types of hill climbing searches become possible.

After we get a solution for the continuous curvature model, we can approximate it by a usual robot manipulator which has a fixed number of rigid links connected by rotational joints (Figure 2). The smoothness of the obtained configurations of continuous curvature model \(^2\) may be helpful here.

The goal of our approach is to build an efficient and robust motion planner which could be applied to on-line control of a manipulator. We hope that our motion planner will prove itself successful in reasonably cluttered space although it will be difficult to specify explicitly for what space configuration we can solve the find-path problem.

1.4 Related Work

Robot motion planning is a very active area of research in AI and Robotics, and the find path problem is a central problem in the area. Basically, there are two kinds of approaches to the problem. Buckley calls them as subdivision algorithms and subdivision-free algorithms in [Buckley, 1989].

Subdivision algorithms are based on subdividing the configuration space [Lozano-Pérez, 1983] into cells and determining the adjacency relationships among the cells. Then, the problem is reduced to checking whether there is a path between the two cells containing the initial and the final configuration. Subdivision algorithms are attractive, because they provide exact algorithmic solutions. But the algorithms are

\(^1\) Although we were inspired by swan's neck movements while observing them in parks, no formal study has been conducted.

\(^2\) They are \(C^2\)-class curves.
intractable in terms of the degree of freedom ([Schwartz and Sharir, 1988] is a survey paper on these algorithms.), thus are not suited to the control of a redundant manipulator at all.

Although motion planning using an idealized manipulator is unique to us, our approach is more related to subdivision-free algorithms ([Peiper, 1988],[Gilbert and Johnson, 1985],[Khatib, 1986],[Buckley, 1989]). In subdivision-free algorithms the problem is considered in the original two or three dimensional space and hill climbing searches are used to get a solution. By remaining in the original space, we can use heuristics in motion planning which could be difficult to think of in the high dimensional configuration space. One elegant subdivision-free algorithm is Khatib's Potential Field Approach. In his paper he claims that his algorithm is efficient enough to be applied to on-line motion planning of a robot manipulator. But subdivision-free algorithms have a severe drawback inherent to their use of hill climbing searches, the local maximum problem [Lozano-Pérez, 1987]. Our approach also employs hill climbing searches and is not free from the local maximum problem. However, redundancy can intentionally be utilized to avoid local maxima in our approach.

1.5 Organization of the Paper

The rest of the paper consists of three parts. In Section 2, we explain the continuous curvature model. In Section 3 we explain the motion planning system for the continuous curvature model. This part is a further development of our initial ideas explained above and has been prepared mainly for us to get a clearer perspective of our research. Some of the ideas presented in the section have not yet been verified by an experiment or a simulation. In Section 4, we show some early results obtained from a simulation.
2 Continuous Curvature Model

The continuous curvature model is an idealized manipulator which has an infinite number of rotational joints. It has been developed to represent a highly redundant manipulator in a compact manner. Its motion is controlled by its curvature. Its configuration is represented by a series of curvature segments. The number of curvature segments is controlled by a decomposition technique to dynamically control the degree of redundancy. Simple configurations can be represented by one curvature segment. More complex configurations are represented by a series of curvature segments.

2.1 Curvature

The curvature function $C(s)$ of a plane curve $[X(s), Y(s)]$ is written as

$$C(s) = \frac{X'(s)Y''(s) - X''(s)Y'(s)}{L^3}$$

where $s$ is a normalized curve length parameter $(0 \leq s \leq 1)$, and $L$ is the length of the curve. For the deduction of the above equation, refer to Section A.1.

Our use of curvature to control a manipulator has good reasons. For planar smooth curves, it is well known that there exists only one curve $[X(s), Y(s)]$ for a given curvature $C(s)$ and a start orientation $[X'(0), Y'(0)]$ ([Stoker, 1969],[do Carmo, 1976] or any book on differential geometry). Moreover, curvature is a coordinate invariant property which is good for reasoning on shapes. In fact the curvature representation of curves has been chosen for many pattern recognition applications (for example, [Kishon and Wolfson, 1987], [O'Rourke and Washington, 1985], [Richards and Hoffman, 1987]).

2.2 Curvature Segment and Curvature Operators

A curvature segment is the basic unit of representation of the continuous curvature model. It is a triple of a length $L$, a start orientation $[X'(0), Y'(0)]$, and a curvature function $C(s)$. $C(s)$ is actually represented by a five point cubic spline function. We knew from experiments that at least five interpolation points are necessary to express relatively complex curves. Please refer to Figure 3. We have the following actions which we call the curvature operators for a curvature segment.

- increase/decrease $C_a, C_b, C_c, C_d, \text{ or } C_e$ (move up/down an interpolation point)
- increase/decrease $S_a, S_b, \text{ or } S_d$ (move right/left an interpolation point)
- rotate the base of a curve

The coordinates of interpolation points $(S_a, C_a, S_b, C_b, \ldots)$ are called curvature parameters. The magnitude of a curvature operator, i.e. an increment/decrement of the curvature parameters is determined in proportion to the distance to a goal.

---

3 A similar theorem holds for 3-D curves with curvature and torsion.

4 When we use the decomposition technique which will be explained, $S_a$ and $S_e$ can also be increased/decreased so that we can change a segment length dynamically.
2.3 Reasoning on Configuration from Curvature

We use the term configuration for the continuous curvature model by analogy with a multi-link manipulator. A configuration of the continuous curvature model is a vector function \([X(s), Y(s), X'(s), Y'(s)]\) which gives us the coordinates and the orientation of all points of the continuous curvature model. The configuration for a given curvature function \(C(s)\) is obtained by solving Equation 1. Since the equation is highly nonlinear and cannot be solved analytically, we solve it numerically. The procedure is described in Section A.2.

Although it is impossible to precisely predict the configuration from a curvature function without solving the curvature equation, we can perform some kinds of qualitative reasonings on configuration from curvature. Let \(r, \theta, \phi\) as defined in Figure 4. Then,

- Radius \(r\) is inversely proportional to \(\int_0^1 |C(s)| ds\).

- End orientation \(\phi\) is proportional to \(\int_0^1 C(s) ds\).

In fact some of the coordinated motions explained in Section 4.4 are developed using this kind of reasoning.

The sign(s) of a curvature function is important in reasoning on the corresponding configuration. If we limit the number of inflection points within a segment to one, we have only five curvature segment types (Figure 5); a straight configuration (type 0), configurations with positive and negative curvature (type 1 and type 2), configurations with an inflection point in the middle (type 3 and type 4). Curvature segment types help to coarsely classify configurations;

- It is possible to choose a suitable curvature segment type without a simulation, given only its base and end orientation.

---

5 We need its initial condition, \([X(0), Y(0)] \) and \([X'(0), Y'(0)]\).

6 An inflection point is a point where the curvature function changes its sign.
Figure 4: Polar Coordinates

Figure 5: Curvature Segment Type
• If the current configuration is type1, and the goal configuration is type2, we readily know that there must be an intermediate configuration which is neither type1 nor type2.

2.4 Decomposition of Segment

The above curvature segment representation alone is not rich enough to express various complex configurations. It is obvious that configurations with two or more inflection points become necessary to achieve a goal while avoiding obstacles in cluttered space. The decomposition technique makes it possible to divide a curvature segment into two or more segments. For a decomposition to be meaningful, we have these decomposition rules;

• The total length of segments generated must be the same as that of the original segment.
• The orientation and the curvature must be continuous at a decomposition point. 

Since the continuous curvature model is built by using continuous functions, we have great flexibility in decompositions;

• We can choose any point as a decomposition point.
• We can move a decomposition point to make one segment longer while making the other shorter.
• If a decomposition becomes unnecessary later, we can merge the segments into one segment.

Since each segment can be controlled quite independently, we can avoid a combinatorial explosion. Note only the continuous model allows this kind of arbitrary, and continuously-changing decomposition we use.

2.5 Approximating Continuous Model with a Multi-link Manipulator

When we finish motion planning for the continuous curvature model, we then try to approximate the obtained plan by a multi-link manipulator. There seems to be many ways for the approximation, and here we introduce one of them (Figure 7). We first group links into pairs of connected links (Link J0-J1, and Link J1-J2 becomes a pair, for example.) Then, even numbered joints are placed on the curve in such a way that they are equidistant on the curve. Positions of odd number joints are automatically determined in this process. We can make a good approximation by putting a limit on the maximum curvature.

7A point where we divide a segment in decomposition is called a decomposition point.
8The decomposition rules are the only constraints.
Figure 6: Decomposition of Segment

Figure 7: Approximation with a Multi-link Manipulator
3 Motion Planning

3.1 Overview

In this section we introduce our motion planning system. Figure 8 shows its overview. The explanation of each component of the system follows. Our design objective is to build a robust planner which works efficiently in a reasonably cluttered environment. We propose four strategies in order not to get caught on local maxima.

- Global motion planning which generates promising plans
- Backtracking to an appropriate previous configuration when a plan fails
- A variety of hill climbing searches to provide a rich planning language
- Adaptive movement control when a manipulator touches an obstacle

3.2 Free Space Representation

We assume that we have the complete knowledge about the environment, where obstacles are expressed as polygons. To represent the free space, we use the generalized cone representation. This representation was used effectively by Brooks for the find path problem of mobile robots [Brooks, 1983]. Our representation is the same as Brooks' except that the base position of a manipulator is also taken into account (Figure 9). The base is regarded as an obstacle when we try to avoid self intersections.

3.3 Global Motion Planning

Global motion planning is carried out in the following sequence.

\footnote{We are still in its design phase. This section has been prepared mainly for us to get a clearer perspective of our research.}

![Motion Planning System Diagram](image-url)
• Hypothesize a final configuration

• Match the final configuration with the current configuration

• Generate a plan to achieve the goal

3.3.1 Hypothesizing a Final Configuration

A final configuration is hypothesized as follows. First we find a navigation path from the base of a manipulator to the goal position (and orientation). This part is easy after we express the free space explicitly. Then the path found (a series of spines of the generalized cones) is converted to a smooth curve using a B spline function ([Gordon and Riesenfeld, 1974],[Foley and Van Dam, 1984]) to get a good approximation of the final configuration (Figure 10). The B spline technique is used because it produces a globally better approximation for the given control points. Other advantages of B spline in hypothesizing a final configuration are its convex hull property which is useful in confining a curve in free space, and its capability to control the smoothness which is useful in reducing the number of the critical points on a curve. We will explain “critical points” shortly.

Generally, we get many candidate configurations, and it is better to evaluate them and find the most promising one. In order to evaluate the configurations, we have these criteria.

• the maximum curvature value

• the number of the critical points on the configuration

• matching with the current configuration

\[10^\text{Nevertheless, further optimization may become necessary to get a satisfactory curve. For example, since it is not possible to specify the length of a curve when we use the B spline technique, we need to adjust the length somehow.}\]
3.3.2 Critical Points

A critical point of a given configuration (curve) \(^{11}\) is defined as follows (also see Figure 11).

- If some point on the curve corresponds to a crossing point from one cone to another and the intersection of the two cones is a small region, then the point becomes a critical point.

- If a cone which contains some part of the curve is slim (like a channel), then the entrance and the exit of the cone become critical points.

- If a curve has more than one inflection point, the second, the fourth, ... are critical points. We need two or more curvature segments to express the curve.

Our intuition is that if we can find enough critical points in advance, then the rest of the problem will be reduced to local hill climbing searches whose subgoals are related to these critical points.

3.3.3 Matching the Final Configuration with the Current Configuration

Next we match the final configuration with the current configuration. There must be some movement between the two configurations (including manipulator end trajectory) which is free from collisions. We match number of features of both configurations for the purpose; the series of cones where they rest, the critical points, the number of segments, and the curvature segment type, ...

Finally the above results are combined to generate a plan which will be executed by the lower level components. Please refer to Figure 12. \(^{12}\)

---

\(^{11}\)In this sub-section we are using the terms "configuration" and "curve" interchangeably.

\(^{12}\)The schemas in the figure are high-level commands to represent subgoals and hill climbing searches to achieve them.
Figure 11: Critical Points

Figure 12: Plan Representation
3.4 Local Movement Control

3.4.1 Hill Climbing Searches

Hill climbing searches are the principle tools for local movement control. In general, hill climbing searches are defined in terms of a set of actions and a heuristic function $h(s)$ where $s$ is a state and $h(s)$ is a distance from the state $s$ to the goal state. In our domain, a curvature operator corresponds to an action. A configuration (equivalently, curvature) corresponds to a state. By setting $h(s)$ appropriately, we get a variety of hill climbing searches to attain many types of goals;

- attain an end position with/without constraining a base orientation
- attain an end position and orientation with/without constraining a base orientation
- attain one of the above goal with a shorter (longer) length
- trace a curve
- move an inflection point without changing an end position/orientation

Moreover, by performing hill climbing searches in parallel using the decomposition technique, we can attain combined or competing subgoals simultaneously;

- reach a point via another point
- pass through a channel
- fold a manipulator
- ...

3.4.2 Adaptive Movement Control after Touching an Obstacle

The subgoal of a hill climbing search is usually specified in terms of the end position/orientation of a segment. When a hill climbing search tries to attain an end position, some other part of the segment may collide with an obstacle. Should we give up the plan and backtrack immediately? No, we can still continue. In our system, touching does not mean a failure of the plan, but a new constraint. After the touching, succeeding movements will be constrained so that the touching point never moves further to the obstacle until it un-touches (see Figure 13). We believe this adaptive control will contribute to the robustness and flexibility of our planning system. Even if we get caught on a local maximum after touching obstacles one after another, we have a way to backtrack. This is another utilization of the redundancy of the continuous curvature model.

13 Touch predicate in Figure 8 informs us if some part of a manipulator touches an obstacle.
3.5 Backtracking

Plans made by the global motion planner are not guaranteed to be collision free. First, our global motion planning is based on a shallow reasoning. It produces a scenario or an assumption rather than a guaranteed plan. Second, we are relying on local movement control which is carried out by hill climbing searches, which may get caught on local maxima. When a hill climbing search fails, we need to backtrack. The principal problem of many planning systems based only on local control strategies is that it is difficult to find an appropriate previous configuration to backtrack. On the other hand, our planning system has two levels, the global planning level and the local control level. We can regard the global motion planner as an assumption generator. Hence we can implement a backtracking technique that will tell us which assumption has failed and thus enables us to backtrack to an appropriate previous configuration. We then try another plan.
4 Experiment with Open Space Problems

After we had designed the outline of our motion planning system as explained in Section 3, we built a simulator and tested our system with some basic problems.

4.1 Open Space Problems

The problems tested are open space problems as opposed to cluttered space problems. In open space problems, there are no obstacles in a manipulator's working environment. Therefore we don't need the decomposition technique, and the number of curvature segments was limited to one.

However, the tested problems consist of some of the important building blocks for the cluttered space problems with decomposition. Table 1 shows the four types of open space problems we tested. OSP1

<table>
<thead>
<tr>
<th>goal to achieve</th>
<th>constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>end position</td>
<td>OSP1</td>
</tr>
<tr>
<td>end position and orientation</td>
<td>OSP3</td>
</tr>
</tbody>
</table>

Table 1: Four Types of Open Space Problems

naturally corresponds to the type of goals to be achieved without decomposition. The rest corresponds to subgoals to be achieved by segments which are generated by decompositions. Please refer to Figure 14. OSP2 corresponds to the type of subgoals and constraints for the last segment, OSP3 for the first segment, and OSP4 for intermediate segments. Only the first segment is allowed to rotate around its starting point (which is actually the base).

Figure 14: Open Space Problems for Decomposed Segments

\[^{14}\text{Segments are numbered from the base to the end.}\]
4.2 Simulator

We built our simulator on a Symbolics machine and name it *Swan's neck simulator* after the actuator which has inspired our research. The simulator handles a neck of one curvature segment and is equipped with basic hill climbing routines to solve the four types of open space problems listed above. Figure 15 shows its simulation window. Displayed in the simulation window are

![Simulation Window](image)

Figure 15: Simulation Window

- a sequence of configurations of a swan's neck.
- a sequence of curvature functions
- a trace table which shows distances to the goal, selected curvature operators, etc.

4.3 Hill Climbing Searches and Their Distance Functions

To solve the open space problems, hill climbing searches are performed in the following sequence.

1. set $d^{curr}$ to the current distance to the goal
2. if $d^{curr} < \epsilon$, exit (the goal has been reached)
3. set $\Delta c$, magnitude of curvature operators in proportion to $d^{curr}$
4. choose the curvature operator which makes the next distance $d^{next}$ the shortest, when applied to the current curvature function $C(s)$
5. if $d^{next} \geq d^{curr}$, exit (we get caught on a local maximum)
6. update $C(s)$ by applying the best curvature operator
7. update the orientation of the base, if the base can rotate
8. goto 1, and repeat

Distance functions are defined for each type of open space problems as follows.

\[
\begin{align*}
    d_{OSP1} &= |\Delta r|/L \\
    d_{OSP2} &= |\Delta r|/L + |\Delta \theta|/\pi \\
    d_{OSP3} &= |\Delta r|/L + |\Delta \theta - \psi|/\pi + |\Delta \phi - \psi|/\pi \\
    d_{OSP4} &= |\Delta r|/L + |\Delta \theta|/\pi + |\Delta \phi|/\pi \\
    \Delta r &= r_{goal} - r \\
    \Delta \theta &= \theta_{goal} - \theta \\
    \Delta \phi &= \phi_{goal} - \phi
\end{align*}
\]

where

- \( L \) : curve length
- \((r, \theta)\) : polar coordinates of the end of the current configuration
- \( \phi \) : orientation of the end of the current configuration
- \((r_{goal}, \theta_{goal})\) : polar coordinates of the goal position
- \( \phi_{goal} \) : goal orientation
- \( \psi \) : rotation of the base, which will be explained shortly

The origin of the polar coordinate system is at the base of the neck (Figure 4). Angles are in radian. When the base is allowed to rotate, its rotation is\(^{15}\)

\[
\psi = \begin{cases} 
\Delta \theta & \text{for OSP1} \\
0.5 \cdot (\Delta \theta + \Delta \phi) & \text{for OSP3}
\end{cases}
\]

The curvature operators defined in Section 2.2 are used as next state functions. In order to make the magnitude of the curvature operators independent from a curve length \( L \), we use a normalized curvature function \( NC(s) \)\(^{10}\) instead of \( C(s) \).

\[
NC(s) = C(s) \cdot (L/2\pi)
\]

Then the magnitude of the curvature operators is set as below based on experiments.

\[
\Delta c = \begin{cases} 
0.20 & \text{if } d^{curr} \geq 0.10 \\
0.10 & \text{if } d^{curr} \geq 0.05 \\
0.05 & \text{otherwise}
\end{cases}
\]

---

\(^{15}\) Rotation is chosen to make the distance the shortest.

\(^{10}\) Curves with \( NC(s) = \pm 1 \) are circles.
Δc is used as an increment/decrement of NC(s) when we move up/down an interpolation point, and 0.5Δc is used as an increment/decrement of s\textsuperscript{17} of NC(s) when we move right/left an interpolation point. Figure 16 shows an example of a successful hill-climbing search. In the right is the sequence of curvature changes for the hill climbing steps. Dots on the graph are points for the cubic spline interpolation. In the left half is the sequence of configurations resulted from the curvature changes. The arrow in the figure shows the goal position and its orientation.

4.4 Local Maximum Problem

OSP1 is easy, because OSP1 has no local maxima. All the others have local maxima. But the local maximum problem is the most evident in OSP4 where achieving r\textsubscript{goal}, \( \dot{\theta} \textsubscript{goal} \), and \( \phi \textsubscript{goal} \) can compete with one another. Hence, we will talk about OSP4 from now on. Figure 17 is a typical example of getting caught on a local maximum. The naive hill climbing search works in Figure 16 but doesn't in Figure 17.

Besides the local maximum problem, the way of the hill climbing search in Figure 17 is not good because it usually leads to solutions with more complex configurations which are either self-intersecting or have large curvature. We prefer such configurations with small curvature, because they are easier to approximate by a multi-link manipulator.

Changing the distance functions, such as putting more weight on Δϕ in d\textsubscript{OSP4} to emphasize an orientation, sometimes works but tends to make another local maximum. We could get rid of some local maxima by using coordinated motions built on top of the original curvature operators. For example increasing \( S_3 \) and decreasing \( S_4 \) at the same time in Figure 3 is effective in decreasing \( r \) without changing \( \phi \) much.\textsuperscript{18} We can thus get an new curvature operator to attain competing goals simultaneously. But in general, coordinated

\textsuperscript{17} s is a normalized curve length parameter and 0 ≤ s ≤ 1.

\textsuperscript{18} Please refer to Section 2.3 to know why.
motions are more effective for the fine tuning of configurations rather than for solving the local maximum problem. We still have local maxima.

4.5 Good Initial Configuration for Hill Climbing Searches

Let's go back to the examples in Figures 16 and 17. Why does the hill climbing search work in one of them and doesn't in the other? In a sense, it is obvious. In the first example, the initial configuration is qualitatively similar to the final configuration. Both are type 3 curvature segments. On the other hand, the initial configuration and the goal configuration are qualitatively different in the second example. We can envision the successful final configuration for the second example as an arc-like one with no inflection point (which is a type 2 curvature segment).

On the basis of the above observation, we added a capability of finding a good initial configuration for hill-climbing searches. We stored in the simulator five typical configurations which are drawn in Figure 18. These configurations represent the five curvature segment types. The maximum of $NC(s)$ for these configurations are set to $\pm 1$.

Before a hill climbing search, the simulator searches the stored configurations for the best initial configuration. The best configuration is the one which is the closest to the goal in terms of the distance functions. Here, we need to have a precompiled movement routine to get from the actual initial configuration from the one suitable for hill climbing. Since we know the curvatures of both configurations, this can be done easily. Figure 19 shows a solution of the same example in Figure 17. This time, the type 2 curvature segment is chosen as the initial configuration for hill climbing.

---

19 Types of curvature segments were explained in Section 2.3.
20 Their curvatures along with $r, \delta$, and $\phi$ are stored.
Figure 18: Five Initial Configurations

Figure 19: Hill Climbing with Good Initial Configuration
4.6 Simulation Results

The initial configuration finding capability has turned out to be very effective. We were afraid that we may need lots of configurations as candidates, but it was not necessary. Judging from our experiments, it seems that the five configurations (and a few more at most) are sufficient as far as open space problems are concerned. The simulator sometimes gets caught on local maxima, but it happens only when the neck is close to the goal. We believe this situation can be improved by adding a few coordinated motions or by fine tuning Δc. We attach here some of the open space problems solved.

Figure 20: Example 1 as OSP1
Figure 21: Example 1 as OSP2

Figure 22: Example 1 as OSP3
Figure 23: Example 1 as OSP4

Figure 24: Example 2 as OSP4
Figure 25: Example 3 as OSP4
5 Summary and Future Work

5.1 Summary
A new approach to motion planning of manipulators was proposed to utilize redundancy in obstacle avoidance. A continuous curvature model was developed to model highly redundant manipulators in a compact and efficient manner. An outline of a motion planning system using the continuous curvature model was presented. Some of the building blocks of the planning system were implemented and then tested with open space problems. The result of the test was satisfactory. On the basis of these encouraging results, we claim that our approach has a significant potential and deserves further research.

5.2 Future Work

5.2.1 Solving Cluttered Space Problems
Our goal is to solve cluttered space problems by using the motion planning system which is sketched in Section 3. The motion planning system will be implemented and tested in a bottom-up fashion. Its basic components, namely the hill climbing routines for a curvature segment, have been finished. Other components will be developed in the following order.

1. Decomposition technique
2. Free space representation
3. Global motion planning
4. ...

In the below, we will talk about some issues which are not covered in previous sections.

5.2.2 Approximating Solutions by a Multi Link Manipulator
The problem of approximating solutions for the continuous curvature model by a multi link manipulator interacts with motion planning for the continuous curvature model. We want to take advantage of our knowledge about a given multi link manipulator \((l_1, l_2, \ldots, l_n)\) where \(l_i\) is the length of the i-th link in the motion planning step for the continuous curvature model. For the purpose, we need to estimate approximation errors in terms of the maximum curvature and the multi link manipulator. On the basis of the estimation, we could put a limit on curvature values to reduce approximation errors, and we could shrink the free space to make sure that the configurations of the multi link manipulator are also collision free.

We could say as the maximum curvature decreases, the approximation error tends to decrease. However, it is unlikely that we can evaluate the approximation errors explicitly as a function of the maximum curvature and \(l_i\)'s (or the number of joints and \(L\), if the joints are equi-distant), considering the nonlinearity of the curvature equation. One way to overcome this would be to use statistics using data gathered by simulations.
5.2.3 Application of Learning Techniques

Our success in solving local maxima problem using good initial-configurations suggests that some learning mechanism is useful in motion planning. However, application of machine learning to geometrical motion planning has been studied very little. It is argued in [Lozano-Pérez, 1987] on the applicability of machine learning to robot programming that

Unfortunately, the crucial motions are not always the largest and most noticeable. Also because of uncertainly and error, what must be learned is often a strategy for relating sensory data to action, a difficult problem in learning.

In [Peiper, 1968], Peiper investigated motion planning of a snake-like digital manipulator, where he suggested the effectiveness of using some kind of learning technique to avoid local maxima. But he didn’t go further than that. In [Ackley, 1988], Ackley applied Neural Network technique to the problem of search and learning inverse kinematics of a six joint planar manipulator. Both Peiper and Ackley are concerned only about open space problems.

We think that learning will be mandatory both for avoiding local maxima during search and for making useful inferences in spite of the high non-linearity of the curvature equation. We agree with Lozano-Pérez that it is difficult to learn crucial motions. We will try to use learning in a modest and controlled way. For finding crucial motions, we will not try to apply learning but the decomposition technique. We are going to learn a simplest kind of cluttered space problem where the manipulator’s working space is restricted within a generalized cone. Let us call the problem swan-in-a-cone problem. What has to be learned here would be the relationship of

- configurations of a unit curvature segment
- curvature
- length of a segment
- size/shape of the cone

We expect that convexity of a generalized cone will enable us to make sound reasoning on the basis of learned knowledge. Our motivation and hope for learning the swan-in-a-cone problem is to transform the problem of motion planning to the one of graph search as is done for mobile robots in [Brooks, 1988].

5.2.4 Formalization

Although this research has sprung from our intuitions, we need a solid formalization so that we can make clear the usefulness and the limitations of our approach. What we are proposing is simply a divide-and-conquer algorithm for motion planning which is made possible by redundancy of manipulators. In order to avoid a combinatorial explosion, we search in the solution space (for the problem where only environment is specified, the manipulator is not specified) with a bias to continuous and smooth configurations. Under the bias, some of the configurations (folding the manipulator like a carpenter’s ruler, for example) is difficult to find. These observations should be formalized in the following framework;

- definition of the search space
• definition of the bias for search
• complexity of the search

It would be nice if we could get clear comparison with the configuration space approach through the formalization.

5.2.5 Long Term Goals

Finally, our long term goals are

• Extending our approach to 3-D space
• Dealing with issues other than geometrical motion planning, such as dynamics and control
• Coping with incomplete information (sensor errors, etc.)

Although an extension of our approach to 3-D space is not trivial, a similar approach would be possible. Since we need torsion as well as curvature to represent a 3-D curve, the continuous model has to be augmented accordingly. Equivalently, we can represent 3-D curves with curvature in X-Z plane and curvature in Y-Z plane which may be more intuitive although the representation is coordinate dependent. The generalized cone representation has to be augmented, too.

5.2.6 Are There Such Manipulators?

In [Yoshikawa, 1984], Yoshikawa used an actual manipulator with seven degrees of freedom (in 3-D) for testing. In order to test our approach which utilizes redundancy more aggressively and is applicable to a more cluttered environment, we will probably need a manipulator with at least nine degrees of freedom for planar problems which is not currently available, although it is interesting to investigate the competence of our approach if it is applied to less redundant manipulators. There is an attempt to design and build a continuous robotic arm [Kokkinis and Wilson, 1988], and we should pay attention to the further development of their research. However, computer simulations are enough when we don’t deal with dynamics.
A Curvature Equation and its Numerical Solution

A.1 Curvature Equation

Curvature function \( C(u) \) of a planar curve \([X(u), Y(u)]\) which is parameterized by an arbitrary parameter \( u \) is defined as:\(^{21}\)

\[
C(u) = \frac{X'(u)Y''(u) - X''(u)Y'(u)}{((X'(u))^2 + (Y'(u))^2)^{3/2}} \quad a \leq u \leq b
\]

It will be convenient to simplify Equation 2 by choosing an appropriate parameter. Let \( L \) be the total length of the curve. When we choose as the parameter \( s \) which is the ratio of the curve length between \([X(a), Y(a)]\) and \([X(u), Y(u)]\) to the total length \( L \), \( s \) satisfies

\[
L \cdot s = \int_0^s \sqrt{(X'(t))^2 + (Y'(t))^2} \, dt
\]

Differentiate the above by \( s \) and we get

\[
L = \sqrt{(X'(s))^2 + (Y'(s))^2}
\]

Then curvature equation is rewritten as

\[
C(s) = \frac{X'(s)Y''(s) - X''(s)Y'(s)}{L^3}
\quad 0 \leq s \leq 1
\]

A.2 Numerical Solution of Curvature Equation

It is known that for a planar curve, curvature \( C(s) \) uniquely determines a curve \([X(s), Y(s)](0 \leq s \leq 1)\) given boundary conditions \([X(0), Y(0)], [X'(0), Y'(0)]\). However, it is impossible to solve the curvature equation (Equation 4) analytically. Instead, we solve it numerically using the following iteration.

initial: \( s = 0 \)
set \( \Delta s \) small enough for the iteration to converge
repeat: compute \([X(s + \Delta s), Y(s + \Delta s)]\) from \([X(s), Y(s)]\) and \([X'(s), Y'(s)]\)
compute \([X'(s + \Delta s), Y'(s + \Delta s)]\) from \([X'(s), Y'(s)]\) and \( C(s) \)
set \( s = s + \Delta s \)
if \( s < 1.0 \) then goto repeat, else exit

It is easy to compute \([X(s + \Delta s), Y(s + \Delta s)]\) from \([X(s), Y(s)]\) and \([X'(s), Y'(s)]\):

\[
X(s + \Delta s) = X(s) + \Delta s \cdot X'(s)
\]

\[
Y(s + \Delta s) = Y(s) + \Delta s \cdot Y'(s)
\]

\(^{21}\)Here we consider sign of curvature instead of taking its absolute value
In order to get \([X'(s + \Delta s), Y'(s + \Delta s)]\), we approximate equation 4 by

\[
X'(s) \cdot \frac{Y'(s + \Delta s) - Y'(s)}{\Delta s} - Y'(s) \cdot \frac{X'(s + \Delta s) - X'(s)}{\Delta s} = L^3 \cdot C(s)
\]  

From equation 3

\[
(X'(s + \Delta s))^2 + (Y'(s + \Delta s))^2 = L^2
\]

By solving Equation 5 and 6 simultaneously, we get \([X'(s + \Delta s), Y'(s + \Delta s)]\) from \([X'(s), Y'(s)]\). There are two algebraic solutions which satisfies eq.5 and eq.6, and the one which is closer to \([X'(s), Y'(s)]\) is what we want. The other one corresponds to the curve with the same curvature but with an opposite orientation.
References


