Toward Planning and Control of Highly Redundant Manipulators*

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

Abstract

There is a need for highly redundant robot manipulators to work in complex, cluttered environments.

We have developped a new and promising approach for the path planning of such manipulators by exploring paths for a completely flexible manipulator : one with an infinite number of degrees of freedom. We call the idealized manipulator a *continuous curvature model*. It is controlled by changing its curvature. The major advantage of the approach is that the complexity of path planning is dependent only on the complexity of the environment, not on the number of degree of freedom of actual manipulators.

We do not need a new architecture to map the solutions obtained for the continous curvature model. We have chosen a manipulator with 12 ball joints as a target hardware architecture and mapped the solution obtained for the continuous curvature model to the manipulator, and its dynamic simulation has been performed.

We believe this is a significant step forward to planning and control of highly redundant manipulators.

1 Introduction

1.1 Use of Redundancy for Obstacle Avoidance

Our goal is to plan motion trajectories for robot manipulators [11]. Intuitively, one would expect the difficulty of trajectory-planning to be a function of the complexity of the environment, and perhaps to become somewhat easier as the manipulator becomes more flexible and more redundant, i.e., has more degrees of freedom. A redundant system allows us to achieve multiple tasks at the same time [20], reaching a position while avoiding obstacles, for example. [8] did research on utilizing redundancy for obstacle avoidance *when a desired manipulator end-effector trajectory is given*. Here we propose a new approach to the problem in which redundancy is utilized more aggressively from the first stage of motion planning.

*This 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 grants IRI-8602665, IRI-8905494, and IRI-8904454, by NASA grants NAG 2-507 and NAG 9-200, and by the Texas Advanced Research Program under grant no. 003658175.

1.2 Our Approach

We propose to explore trajectory-planning for a *continous curvature manipulator*. The continous curvature manipulator is a completely flexible manipulator: one with an infinite number of degrees of freedom. It is controlled by perturbing the function specifying the continuously-changing curvature along the length of the manipulator. A set of hill-climbing routines are adequate for achieving a basic set of goals within an unobstructed space. When the space is obstructed, the manipulator can be segmented at points determined by the structure of the environment, and individual open-space routines are combined, along the length of the manipulator, to achieve a solution to the more complex problem.

The major advantage of the continuous approach is that spatial problem-solving takes place in the low-dimensional physical workspace, rather than in a high-dimensional configuration space. If we assign unit cost to the hill-climbing algorithm for solving any open-space problem, then the complexity of trajectory planning is dependent only on the complexity of the environment.

Once a trajectory has been created for the continuous manipulator, the solution may be mapped back onto a jointed arm. This mapping, of course, becomes *easier* as the jointed arm has more degrees of freedom.

1.3 Related Work

There are two kinds of approaches to manipulator motion planning, *subdivision* algorithms and *subdivision-free* algorithms [2]. Subdivision algorithms are based on subdividing the *configuration space* [12] into cells and determining the adjacency relationships among the cells. Subdivision algorithms are intractable in terms of the degree of freedom [18], thus are not suited to the control of a redundant manipulator. In subdivision-free algorithms ([16],[7]), the problem is considered in the original two or three dimensional space and hill climbing searches are used to get a solution. Subdivision-free algorithms have a severe drawback inherent in their use of hill climbing searches: the *local maximum problem*. Our approach also employs hill climbing searches, thus is not free from the local maximum problem. However, redundancy can *intentionally* be utilized to avoid local maxima in our approach.

2 Continuous Curvature Manipulator

The continuous curvature manipulator 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.

2.1 Curvature Segment and Curvature Operators

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

$$C(s) = \frac{X'(s)Y''(s) - X''(s)Y'(s)}{L^3} \tag{1}$$

where s is a normalized curve length parameter ($0 \leq s \leq 1$), and L is the length of the curve.

A curvature segment is the basic unit of representation of the continuous curvature manipulator. 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 (Figure 1). We have the following *curvature operators* for a curvature segment.

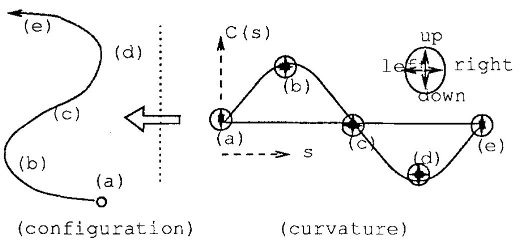


Figure 1: Curvature Segment Representation and its Operators

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

A configuration of the continuous curvature manipulator 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 manipulator. The configuration for a given curvature function $C(s)$ is obtained by solving Equation (1) with its initial condition, $[X(0), Y(0)]$ and $[X'(0), Y'(0)]$ numerically [11].

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 2). 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 position/orientation.

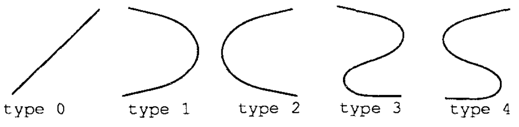


Figure 2: Curvature Segment Type

2.2 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. 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. Note these decompositions exploit the continuity of the model, and the fact that the segment boundaries can be moved smoothly along the length of the arm.

3 Open Space Problem

The open space problem is attempting to reach a goal when there are no obstacles in a manipulator's working environment. We do not need the decomposition technique for this. The open space problem can be seen as *inverse kinematics* of a curvature segment.

3.1 Hill Climbing Search

A set of hill-climbing routines are adequate for open space problems. Distance functions are defined based on the goal, the tip and the rotation of the base [11]. The curvature operators are used as next state functions. The left half of Figure 3 shows an example of a successful hill-climbing search.

3.2 Solving Local Maximum Problem

The right half of Figure 3 is a typical example of getting caught on a local maximum. The naive hill climbing search does not work here, because the initial configuration and the goal configuration are qualitatively different. Hence, we added a capability of finding a good curvature segment type in Figure 2. Before a hill climbing search, the simulator searches for the best curvature type. Figure 4 shows a solution to the example. The initial configuration finding capability has

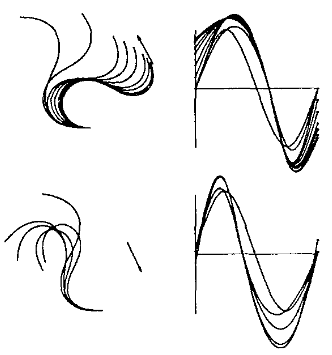


Figure 3: Successful Hill-Climbing and Local Maximum in Hill Climbing. Each example contains a sequence of curvature changes for the hill climbing steps (in the right) and the sequence of configurations resulting from the curvature changes (in the left). The arrow in the figure shows the goal position and its orientation.

turned out to be very effective. Judging from our experiments, it seems that the five curvature types are sufficient.

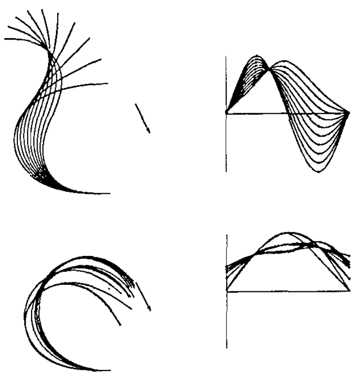


Figure 4: Hill Climbing with Good Initial Configuration

4 Using Decomposition for Obstacle Avoidance

When the space is obstructed, the manipulator can be segmented at points determined by the structure of the environment (*decomposition technique*), and individual open-space routines (which are called *schemas*) are combined, along the length of the manipulator, to achieve a solution to the more complex problem.

4.1 Our Approach to Obstacle Avoidance

To extend our results for open space problems to cluttered space problems, we need
(1) Decompose Free space into convex regions.
(2) Find subgoals to reach the goal while keeping the manipulator within the convex regions.
(3) Decompose a continuous curvature manipulator for conjunctive subgoals.
(4) Hill climbing search to achieve each subgoal.
There are many methods to decompose free space into convex regions ([1], [10],[17], for example), and once decomposition is finished, subgoals can be found by searching the connectivity graph of convex free regions for a path to reach a goal. For traversing within a region, we can use a straight line segment. For making a smooth turn between regions, we can use a cubic spiral curve which has 0 curvature at both ends [6].

To achieve conjunctive goals, we use decomosition. Four decompositions have been implemented: *add/delete* a segment, *divide* a segment at a specified point, and *merge* two neighboring segments. Each subgoal is achieved by executing schemas which are generalizations of pure hill climbing searches to model various kinds of *local* movements.

4.2 Schemas

To solve the local maximum problem in open space (Figure 4), we used three schemas. (in the sequence of SCM-1,SCM-2, and SCM-0).
[SCM-0] Pure hill-climbing to goal: tip position and/or orientation.
[SCM-1] Select appropriate curvature segment type (Figure 2) and base rotation for motion schema SCM-2 or SCM-8. (Does not move the arm.)
[SCM-2] Move by interpolation from current curvature to one selected by SCM-1.

To solve cluttered space problems where we need to use decomposition technique, we added the following schemas.
[SCM-3] Maintain current tip position/orientation while increasing or decreasing the length allocated to a segment (i.e. Feed or Retract).
[SCM-4] Same as SCM-1, but also determine optimal segment length to achieve goal.
[SCM-5] Feed or retract a segment along a given curve, specified by its curvature.
[SCM-6] Change curvature at end to zero, while preserving tip position/orientation. (For interface between segments.)
[SCM-7] Sequence of SCM-0, with subgoals chosen to ensure straight-line tip motion.
[SCM-8] SCM-2 while preserving direction of vector from base to tip.

4.3 Using Decomposition and Schemas

Here we demonstrate the use of decompositions and schemas implemented so far. Figure 5 is a trace output from the simulator. In this example, a heuristic for obstacle avoidance in [16] : fold the manipulator to shorten it and then move it in front of any obstacle, was used in lieu of free space decomposition and finding subgoals procedures. The subgoal is given as a midpoint between the base and the obstacle and has an orientation perpendicular to the line between the base and the midpoint.

5 A Dynamic Simulation of Swan's Neck

Although there are attempts to design and build continuous manipulators ([19], [9]), we do not need a new architecture to map the solutions obtained for the continuous curvature model. We have done a design of a 36 D.O.F. manipulator (which we call *swan's neck* manipulator), a mapping of the trajectory to swan's neck, its inverse kinematics, trajectory planning, dynamic modeling, and near real-time drawing on 3D space window of Silicon Graphics machine.

5.1 Overview of Dynamic Simulation

It is sometimes more difficult to control redundant manipulators. One of common manipulator control is done using inverse Jacobian matrix J^{-1}. But J^{-1} does not exist for redundant manipulators. Hence use of pseudo inverse Jacobian matrix $J^{\#}$ is proposed [3], but this is not easy because it is necessary to include obstacle avoidance configuration in $J^{\#}$ at velocity level. Instead, we first determine joint angle velocities by trajectory planning in joint space. This is done by mapping the collision free trajectory obtained for the continuous curvature manipulator to swan's neck. The velocities in work space may then be calculated using J.

The computation step is as follows:
(1) Open window for drawing.
(2) Get next intermediate configuration to pass.
(3) Do inverse kinematics for above configuration.
(4) Do trajectory planning in joint space.
(5) Do dynamic simulation.
(6) Draw swan's neck.
(7) If arrived at (in fact near) intermediate destination then go to step 2, else go to step 5 for next time interval.

5.2 Swan's Neck Manipulator and its Inverse Kinematics

We assumed to use ball joints. For each ball joint, 3 D.O.F. are allowed. The designed manipulator has total 12 ball joints and 36 D.O.F. The kinematic notations used here follow those of [15]. One segment of swan's neck and its corresponding link parameter table is shown in Figure 6.

Since the current planning for the continuous curvature manipulator is done in 2D space, $a_y = n_y = p_y = 0$, $\underline{o} = (\ 0\ 1\ 0\ )^T$ where $\underline{a}$ is approach vector, $\underline{o}$ is orientation vector and $\underline{n} = \underline{o} \times \underline{a}$. Subscript y means y direction of the vector. Here approach vector is designed to be the same direction as that of the beak of a swan. By calculating inverse kinematics of this one segment manipulator, we get

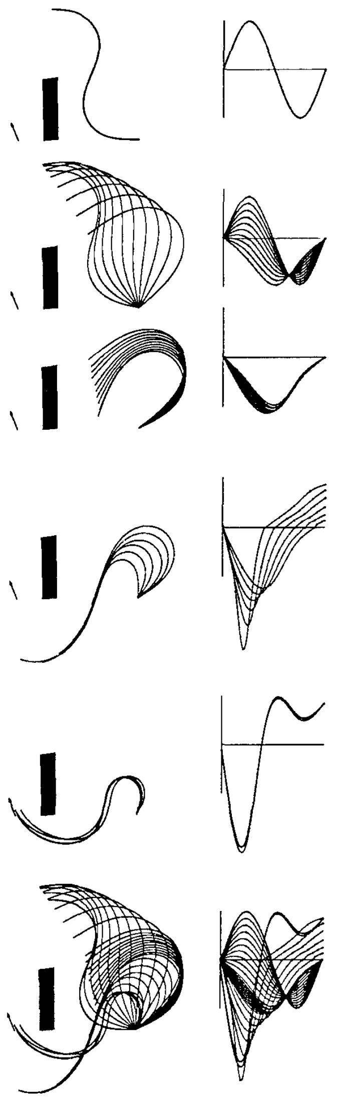
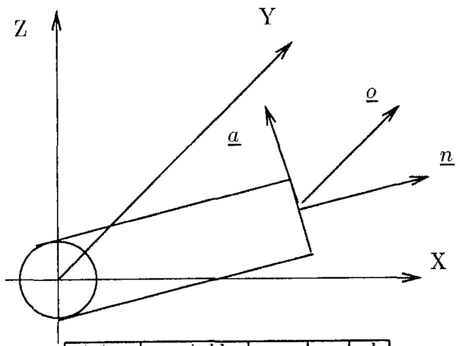


Figure 5: Using Decompositions and Schemas: The plan is expressed as the following steps. (1) The initial configuration and the goal. (2) Change the curvature segment type to achieve the subgoal [SCM-1],[SCM-8]. (3) Proceed straight to the subgoal [SCM-7]. (4) Change the end curvature to zero [SCM-6]. Add a segment at the tip. The base segment tries to keep the subgoal with a shorter length [SCM-3]. The tip segment traces the curve to reach the goal from the subgoal [SCM-4],[SCM-5]. (5) Merge the segments when the length of the base segment becomes short enough. Hill climbing to the goal [SCM-0]. (6) All the movements included.



joint	variable	α	a	d
1	θ_1	-90	0	0
2	θ_2	90	0	0
3	θ_3	0	h	0

Figure 6: One Segment of Swan's Neck and its Link Parameter Table

$$\theta_2 = tan^{-1}(\frac{-P_y}{P_x}), \quad \theta_1 = \theta_3 = 0 \ .$$

By the mapping to be explained shortly, we can get every joint positions and do inverse kinematics for each segment, then we get all joint values for swan's neck manipulator.

5.3 Mapping to a Jointed Arm

Let us show a mapping to a jointed arm which has an even number $2n$ of links of the same length. 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 equi-distant on the curve. Positions of odd number joints are automatically determined in this process. Using this mapping for swan's neck, the solution in Figure 5 is mapped to Figure 7.

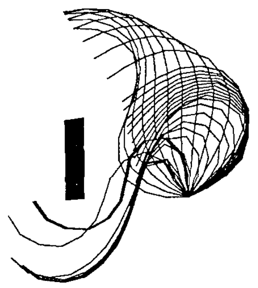


Figure 7: Mapped Trajectory

5.4 Dynamic Simulation

A segment of swan's neck is defined as cylinder of length h and radius r, which has ball joint at the left end of a segment, and i th coordinate at the right end. In computing the inertia tensor, we assume that the distribution of mass over the neck is the same. We neglect the mass of a ball joint.

The dynamic simulation of swan's neck was done using Newton-Euler formulation [1]. One merit of Newton-Euler formulation is that computational complexity is linear according to the degree of freedom ([5], [13]). Trajectory planning was done in joint space not in Cartesian space of end-effector. Our trajectory planning method is exactly the same as that of [15].

Figure 8-10 is a result of a simulation, given the path in Figure 7. In the simulation we assumed m_i = 0.1 Kg if i is divisible by 3 , otherwise m_i = 0 Kg where i = 1 , $\cdots$, n. Also h = 2r = 0.05 m. Dynamic calculation and drawing on Silicon Graphics machine is done about 5 times per second.

6 Summary

A new approach to motion planning of manipulators has been developed to utilize redundancy in obstacle avoidance. The continuous curvature manipulator was defined as an idealization of highly redundant manipulators, and its direct and inverse kinematics were developed. Decompositions and schemas were implemented, and their use for obstacle avoidance planning was demonstrated. The trajectory obtained was then mapped to the 36 D.O.F. manipulator and its dynamic simulation was successfully performed.

The path planning problem for highly redundant manipulators is getting more attention. Recently, Chirikjian and Burdick have developed a similar approach independently [4]. In particular, they have found a closed form solution of inverse kinematics for a particular class of curvature functions, while we use hill climbing search with more general 5 point spline curvature functions.

We plan to complete the path planning as sketched in Section 4.1.

References

[1] R. A. Brooks. Solving the find-path problem by good representation of free space. *IEEE transaction on Systems, Man and Cybernetics*, 13:190–197, 1983.

[2] C. E. Buckley. A foundation for the flexible-trajectory approach to numeric path planning. *The International Journal of Robotics Research*, 8(3):44–64, 1989.

[3] P. H. Chang. A closed-form solution for inverse kinematics of robot manipulators with redundancy. *IEEE Journal of Robotics and Automation*, RA-3(5), 1987.

[4] G. S. Chirikjian and J. W. Burdick. An obstacle avoidance algorithm for hyper-redundant manipulators. In *Proc. IEEE International Conference on Robotics and Automation*, pages 625–631, 1990.

[5] J. J. Craig. *Inroduction to Robotics: Mechanics and Control.* Addison-Wesley, 1986.

[1]Because of the length limitation, we can not include the equations. They are included in [14].

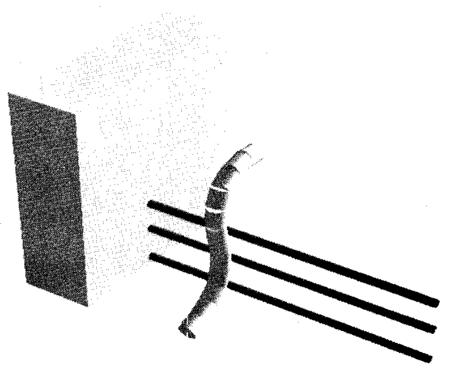


Figure 8: Dynamic Simulation (1)

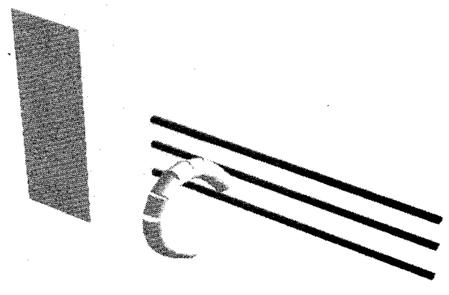


Figure 9: Dynamic Simulation (2)

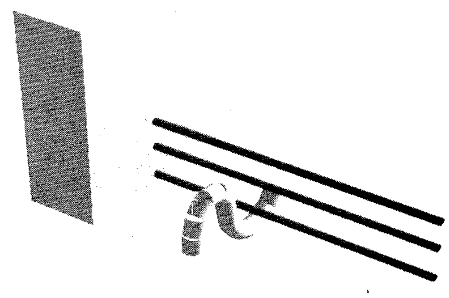


Figure 10: Dynamic Simulation (3)

[6] Y. Kanayama and B. I. Hartman. Smooth local path planning for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1265–1270. IEEE, 1989.

[7] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotic Research*, 5(1):90–98, 1986.

[8] M. Kirćanski and M. Vukobratović. Contribution to control of redundant robotic manipulators in an environment with obstacles. *The International Journal of Robotics Research*, 5(4):112–119, 1986.

[9] T. Kokkinis and J. D. Wilson. Design of continuous robotic arms. In M. Jamshidi, J. Y. S. Luh, H. Seraji, and G. P. Starr, editors, *Robotics and Manufacturing.* ASME Press, 1988.

[10] D. T. Kuan, J. C. Zamiska, and R. A. Brooks. Natural decomposition of free space for path planning. In *Proc. IEEE International Conference on Robotics and Automation*, pages 178-183, 1985.

[11] B. J. Kuipers and A. Hayashi. Geometrical motion planning of manipulators using a continous curvature model. Technical Report AI89-114, Artificial Intelligence Laboratory, The University of Texas at Austin, September 1989.

[12] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[13] J. Y. Luh, M. W. Walker, and R. P. Paul. On-line computational scheme for mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3), 1980.

[14] J. Park. Dynamic simulation of swan's neck. Unpublished Manuscript, March 1990.

[15] R. P. Paul. *Robot Manipulators: Mathematics, Programming and Control.* MIT Press, 1981.

[16] D. L. Peiper. *The kinematics of manipulators under computer control.* PhD thesis, Stanford University, Mechanical Engineering Department, 1968.

[17] K. D. Rueb and A. K. C. Wong. Structuring free space as a hypergraph for roving robot path planning and navigation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 9(2):263–273, 1987.

[18] J. T. Schwartz and M. Sharir. A survey of motion planning and related geometric algorithms. *Artificial Intelligence*, 37:157–169, 1988.

[19] D. J. Todd. *Fundamentals of robot technology.* John Wiley and Sons, 1986.

[20] T. Yoshikawa. Analysis and control of robot manipulators with redundancy. In M. Brady and R. Paul, editors, *Robotics research: the first international symposium*, MIT Press series in artificial intelligence. MIT Press, 1984.