In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*
Cambridge, MA: AAAI/MIT Press, 1991.

Path Planning for Highly Redundant Manipulators using a Continuous Model*

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

Abstract

There is a need for highly redundant manipulators to work in complex, cluttered environments. Our goal is to plan paths for such manipulators efficiently.

The path planning problem has been shown to be $PSPACE$-complete in terms of the number of degrees of freedom (DOF) of the manipulator. We present a method which overcomes the complexity with a strong heuristic: utilizing redundancy by means of a continuous manipulator model. The continuous model allows us to change the complexity of the problem from a function of both the DOF of the manipulator (believed to be exponential) and the complexity of the environment (polynomial), to a polynomial function of the complexity of the environment only.

Introduction

Highly Redundant Manipulators

Redundant manipulators have more degrees of freedom (DOF) than necessary for a specified class of tasks. There is a need for highly redundant manipulators to work in complex, cluttered environments. Their applications include passing trough restricted passages for the inspection or the maintenance of a mechanical system such as a nuclear reactor and a spacecraft.

In the literature, highly redundant manipulators have been given a variety of names including ORM (the Norwegian word for snakes) [Pieper 68], elastic manipulator [Hirose *et al.* 83], spine robot [Drozda 84, Todd 86], tentacle manipulator [Ivanescu and Badea 84], elephant's trunk like elastic manipulator [Morecki *et al.* 87], snake-like manipulator [Clement and Iñigo 90]. Some were actually built. While many of them are so called continuous arms, highly articulated arms are also studied.

Although much work has been done on the study of mechanical designs for highly redundant manipulators, little attention has been paid to kinematics and path planning for such manipulators.

Path Planning Utilizing Redundancy

The path planning problem is the problem of finding a collision free trajectory for a manipulator between an initial state and a goal state, when its environment is known. Path planning is an important component of task level programming [Lozano-Pérez 83a]. Humans seem to be good at path planning with their arms, and we believe that the performance is attributed largely to the kinematic redundancy of our arm-body system. However, no path planning algorithm exists to utilize redundancy.

Path planning algorithms based on the configuration space approach [Lozano-Pérez 83b] are intractable in terms of the number of DOF. Algorithms based on the artificial potential field approach [Khatib 86] are more computationally feasible, but have a drawback inherent in their use of local optimization techniques: the *local minima problem*. Previous research on explicitly utilizing redundancy for obstacle avoidance is limited to controlling a manipulator when its end effector trajectory is given.

Continuous Manipulator Model

We explore kinematics and path planning for highly redundant manipulators by means of a *continuous manipulator model*. The shape of continuous arms along its center line can be directly expressed by the continuous model. Even for jointed arms, their macroscopic shape can be expressed. The continuous manipulator model is controlled by continuously-changing curvature and torsion, intrinsic properties of smooth curves, along the length s of the manipulator.

The continuous model in 2-D is controlled by its curvature κ. A segment is the basic unit of representation for the continuous model. For each segment, its curvature function $\kappa(s)$ is discretized using five points in the curvature graph. To change the shape of the segment,

*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-8905494 and IRI-8904454, by NASA grant NAG 2-507, and by the Texas Advanced Research Program under grant no. 003658-175.

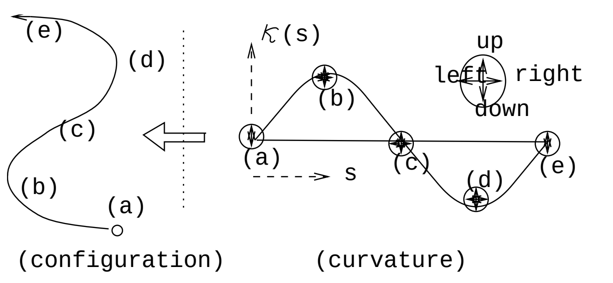

Figure 1: Curvature Segment Representation and its Operators. The following *curvature operators* are used to change curvature (and configuration). **a.** Increase/decrease κ_a, κ_b, κ_c, κ_d, or κ_e. **b.** Increase/decrease s_b, s_c, or s_d. **c.** Rotate the base.

curvature operators are defined to move the points. See Fig. 1.

The continuous model in 3-D is controlled by both curvature κ and torsion τ. For each segment, its torsion function $\tau(s)$ is also discretized using five points (s_a, τ_a), (s_b, τ_b), (s_c, τ_c), (s_d, τ_d), and (s_e, τ_e). Operators now include those to move (s_a, τ_a) through (s_e, τ_e). We use the Frenet equations (1) to obtain a configuration from curvature and torsion (see [Stoker 69]).

$$\begin{pmatrix} \dot{\mathbf{v}}_1(s) \\ \dot{\mathbf{v}}_2(s) \\ \dot{\mathbf{v}}_3(s) \end{pmatrix} = \begin{pmatrix} 0 & +\kappa(s) & 0 \\ -\kappa(s) & 0 & +\tau(s) \\ 0 & -\tau(s) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1(s) \\ \mathbf{v}_2(s) \\ \mathbf{v}_3(s) \end{pmatrix} \quad (1)$$

$\mathbf{v}_1(s)$, $\mathbf{v}_2(s)$, $\mathbf{v}_3(s)$ are the tangent, normal, and binormal vectors. After we obtain $\mathbf{v}_1(s)$ by integrating (1) numerically, the configuration $\mathbf{P}(s) = (x(s), y(s), z(s))$ is obtained using

$$\mathbf{P}(s) = \mathbf{P}(s_0) + \int_{s_0}^{s} \mathbf{v}_1(\sigma) d\sigma$$

The number of segments is controlled by a decomposition technique to dynamically change the degree of redundancy. For a decomposition to be meaningful, we have the following decomposition rules;

- The total length of segments generated must be the same as that of the original segment.
- Curvature/torsion and orientation must be continuous at a decomposition point.

Because of the continuity of our model, we have great flexibility in decompositions. In particular, we can choose any point as a decomposition point, and we can move a decomposition point smoothly along the length of the continuous model to make one segment longer while making the other shorter.

Our Approach to Path Planning

First, we develop motion schemas for the individual segments to achieve a basic set of goals in open and cluttered space. Second, we plan a smooth trajectory through free space for the *end effector* with a maximum curvature constraint, by searching a connectivity graph of primary convex regions. Third, the trajectory generates a set of position subgoals for the continuous manipulator which are achieved by the basic motion schemas. Fourth, the mapping from the continuous model to the available jointed arm provides the curvature bound and obstacle envelopes required (in step 2) to guarantee a collision-free path.

[Chirikjian and Burdick 90] presents an approach similar to ours. While we use 5 point interpolation to discretize curvature and torsion, they use a modal decomposition. However, in their paper, obstacle avoidance was accomplished by manual decomposition and selection of curvature functions. Also, the problem of bounding the error in the mapping from the continuous model to the jointed arm is not addressed.

The Basic Motion Schemas

Following is the list of basic motion schemas for an individual segment to achieve a basic set of goals in open and cluttered space.

Hill-climb: Hill climbing search to achieve tip position/orientation or end curvature/torsion. The curvature/torsion operators are used as next-state functions.

Interpolate: Move by interpolation between two specified curvature/torsion profiles.

Feed/Retract: Increase (decrease) the length allocated to a segment by moving the tip along a trajectory to reach a given position/orientation. This is a motion schema to represent the follow-the-leader type, snake-like motion considered in [Clement and Iñigo 90].

Fold/Unfold: Increase (decrease) the length allocated to a segment, while maintaining tip position and orientation.

Open Space

Hill-climb and *Interpolate* schemas are used to achieve position/orientation in open space. The naive hill climbing search works when its initial state is close to a goal state (Fig. 2), but does not work in the example in Fig. 3. To eliminate the problem, we add a capability of selecting and interpolating to a good initial configuration before hill-climbing search (Fig. 5). Five curvature segment types in Fig. 4 are used as candidates for good initial configurations in 2-D.

Cluttered Space

We assume there is enough open space around the base to fold the manipulator. To achieve a position in cluttered space, the continuous manipulator is retracted, rotated, and then extended. Once subgoals along a path are obtained, segments are added one by one at

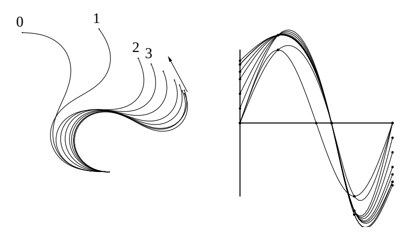

Figure 2: Successful Hill-Climbing. This and subsequent figures show graphical output from our simulator. Each display shows multiple plots of $(x(s), y(s))$ on the left and $\kappa(s)$ on the right for a finite sequence of times $t_0, \ldots, t_n$. The arrow in the figure shows the goal position and its orientation.

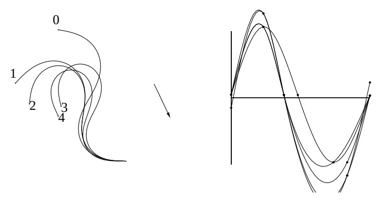

Figure 3: Local Minimum in Hill Climbing

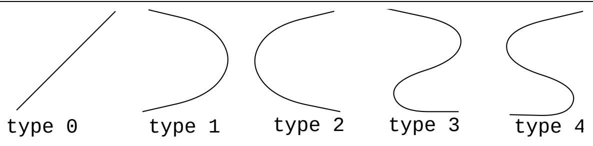

Figure 4: Curvature Segment Type

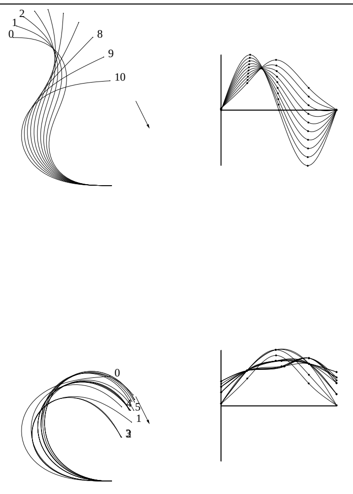

Figure 5: **(Top)** Interpolate and **(Bottom)** Hill Climb: move by interpolation from the initial state to the instance nearest to the goal, then hill climb to the goal.

the tip to achieve each of the subgoals. *Feed* schema is used with the tip segment to achieve a subgoal, while *Unfold* schema is used with the segment folded as a circular arc to provide the length to *Feed* to provide the length to *Feed* the tip segment along a trajectory (see Fig. 6). The trajectories between subgoals are cubic spiral curves which will be explained shortly.

Planning a Smooth path for a Point Robot

We present an algorithm to find a smooth path (i.e. a continuous curvature path) under a maximum curvature constraint *for a point robot.* An end effector trajectory for the continuous manipulator is obtained by the algorithm. Our algorithm first decomposes free space into convex regions. Then, smooth paths are found by extending previous algorithms to find polygonal paths in the convex regions. Overlapping regions of the convex regions are used to make smooth turns from one region to another.

Free Space Decomposition to PCRs

Free space is decomposed into *primary convex regions* [Rueb and Wong 87]. A primary convex region (PCR) is an unobstructed convex region with each boundary edge covering some portion of an obstacle wall. See Fig. 7. In the figure, each region is shrunk only for visibility. PCRs are found efficiently by a directed search for a set of fundamental circuits in an abstract graphical representation of the environment geometry.

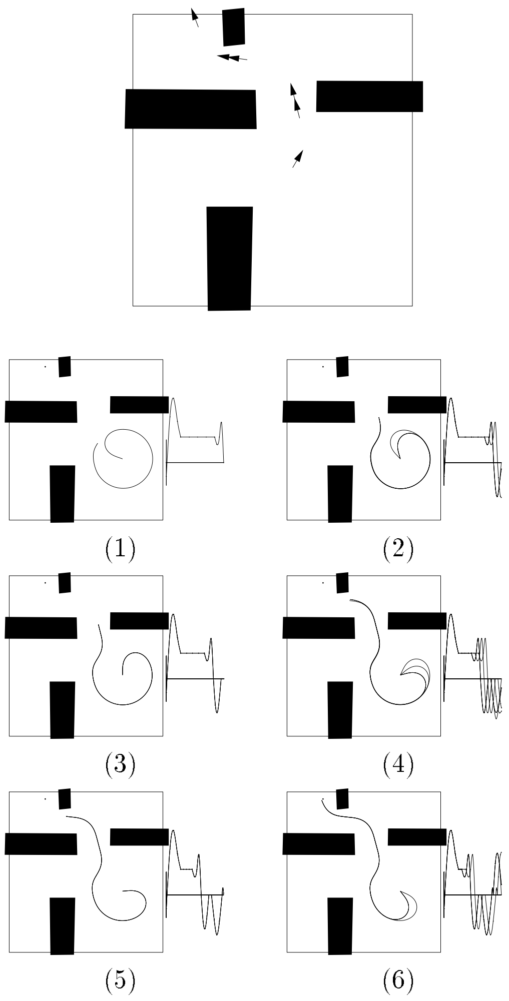

Figure 6: Achieving Subgoals along a path

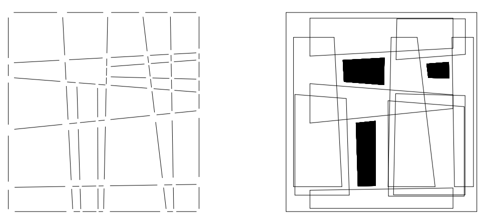

Figure 7: Wall segments and PCRs

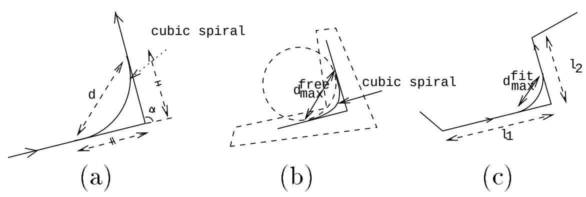

Figure 8: **(a)** Smooth Turn using a Cubic Spiral **(b)** d^{free}_{max} **(c)** d^{fit}_{max}

Making a Smooth Turn between PCRs

We need to locate *turning corners* appropriately, both to make a smooth turn from one PCR to another while satisfying the maximum curvature constraint, and to find a shorter path to reach a goal. For a small overlapping region, we use its center of gravity as a turning corner. A large overlapping region can be further divided around its center of gravity in order not to miss smooth turns.

[Kanayama and Hartman 89] presents a method to make a smooth move from one position and orientation to another, using cubic spiral curves. We use cubic spirals to provide a continuous curvature path, since they can be constructed to have zero curvature at tangent points. A cubic spiral is a curve whose orientation (integration of curvature) is described by a cubic function of path distance s.

Proposition 1 (Kanayama and Hartman) *If the size d and the deflection α of a cubic spiral is given (Fig. 8a), its length l and curvature κ are*

$$l = \frac{d}{D(\alpha)} \tag{2}$$

$$\kappa(s) = \frac{6\alpha D(\alpha)^3}{d^3}((\frac{l}{2})^2 - s^2) \tag{3}$$

where $s \in [-\frac{l}{2}, +\frac{l}{2}]$ and $D(\alpha) = 2\int_0^{1/2}\cos(\alpha(3/2 - 2s^2)s)ds$.

This result is directly applicable to making a smooth turn. For each candidate corner, we check whether we can make a turn as follows.

1. Find d_{min}, the minimum d consistent with the maximum curvature constraint.
2. Find d^{free}_{max}, the maximum d for which the curve lies entirely within free space.
3. Find d^{fit}_{max}, the maximum d for a cubic spiral to fit along both tangent line segments.
4. Check $d_{min} \leq min(d^{free}_{max}, d^{fit}_{max})$. This guarantees that we can make a collision free turn within the maximum curvature.

In order to find d_{min}, note that $\kappa(s)$ in (3) has its maximum at the midpoint. Hence, $d \geq d_{min} = \frac{1.5\alpha D(\alpha)}{\kappa_{max}}$.

To find d_{max}^{free}, we notice that a cubic spirals is always contained in the area outlined by its tangent lines and the circular arc which is tangent at the same points. To find a tangent arc which is both collision free and has the maximal radius r_{max}, we apply the condition that the arc passes through one of the corners of the overlapping region (Fig. 8b). From this, we obtain $d \leq d_{max}^{free} = 2r_{max}sin(\alpha/2)$.

It is possible to find whether we can fit smooth turns by using d_{min} and d_{max}^{free} obtained, *given a whole candidate polygonal path*. However, this leads to an exhaustive search. We use a local fit method instead. When making a turn, we confine its starting and ending points within the distance of $l_{min} = min(l_1/2, l_2/2)$ from the turning corner, where l_1 (l_2) is the length of a incoming (outgoing) line segment (Fig. 8c). To make a turn within l_{min}, we require that $d \leq d_{max}^{fit} = 2l_{min}cos(\alpha/2)$.

Graph Search

We now build a connectivity graph and search for a path which satisfies the maximum curvature constraint. Nodes in the connectivity graph represent the straight line segments within PCRs. An edge from a node N_i to N_j exists if and only if the corresponding line segments L_i and L_j share an end point and there is a smooth turn from L_i to L_j as explained in Section . We use the A^* algorithm to find a path in the connectivity graph. As a heuristic function, we use Euclidean distance from a current node (midpoint of its line segment) to a goal position. Fig. 9 shows the steps involved in the path planning. Fig. 10 shows the paths found.

Complexity

A loose upper bound on the complexity of our algorithm is obtained as follows. An upper bound on the algorithm to find PCRs [Rueb and Wong 87] is $O(n^4)$ time in the number of obstacle edges. If we treat the number of candidate turning corners as a constant, the number of nodes for A^* search is bounded by $O(n^{12})$. This is because there cannot be more than $O(n^4)$ PCRs and a node in the graph is determined by a sequence of 3 PCRs. By using the algorithm in [Martelli 77] to improve the exponential worst case running time of A^*, we obtain $O(n^{24})$ as an upper bound.

The algorithm is quite efficient in practice. Rueb and Wong have also reported an $O(n)$ performance result for his experiment as opposed to the $O(n^4)$ upper bound. This immediately makes our algorithm run in $O(n^6)$ time instead of $O(n^{24})$. Furthermore, average case running time for A^* search is much better because of the heuristic associated. In fact, the four paths shown in Fig. 10 were found in 11, 18, and 119 seconds respectively on a Symbolics 3670 without floating point hardware.

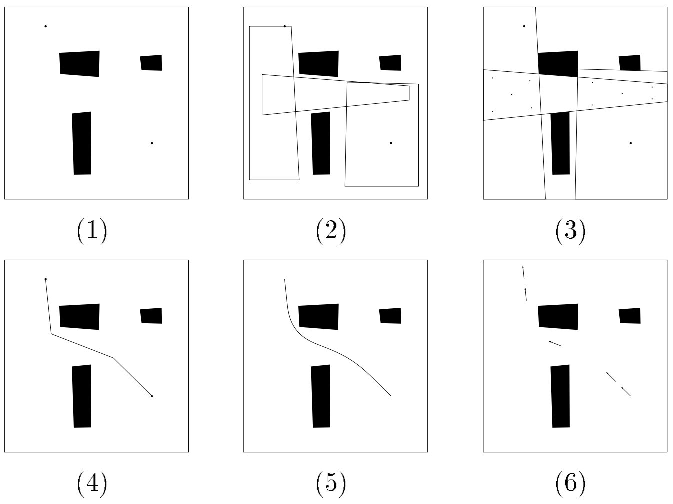

Figure 9: Steps Involved in Path Planning. **(1)** Initial and goal position is given. **(2)** Identify PCRs. (only those on the solution path are shown.) **(3)** Identify candidate turning points in overlap regions. **(4)** Find least cost path in connectivity graph, consistent with maximum curvature constraint. **(5)** Create smooth path by inserting cubic spirals. **(6)** Identify subgoals as start/end points of turns of the path.

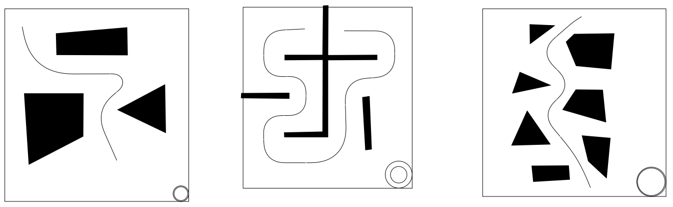

Figure 10: Paths Found. Inner circles at the bottom right have the maximum curvature given for searches, and outer circles have maximum curvature for the paths found (radius is the inverse of curvature). In these examples, only the centers of gravity of overlaps are used as candidate turning corners.

Finding Subgoals for Manipulator

First, locate the folded manipulator. The primary convex region which contains the folded manipulator is called the *base PCR*. When we fold the manipulator as a circular arc, we can extend the manipulator from anywhere on the circle by rotating around the base. Hence, as initial states of the graph search, we use tangent lines to the circle from all candidate turning corners in the overlaps with the base PCR. These initial states naturally correspond to partial paths through which we can extend the manipulator. After defining the initial states, graph search proceeds exactly in the same manner to generate the position subgoals in Fig. 6. The modification does not change the complexity of the original algorithm to find smooth paths.

Extend to 3-D

Two approaches are feasible for path planning in 3-D space.

[Brooks 83a] proposed decomposing free space into generalized cones in order to find a path for mobile robots. The same free space representation was then used to plan a collision free path for manipulators by restricting the hand movement [Brooks 83b]. Free space in 3-D is represented by its horizontal 2-D slices. With this $2\frac{1}{2}$-D approach, most of the method we have explained can be used without modification.

Alternatively, we decompose 3-D free space into primary convex regions. Smooth 3-D curves with curvature and torsion will be used in lieu of cubic spirals to make turns from one such region to another. We (see [Hayashi 91]) have extended the algorithm in [Singh and Wagh 87] to find primary convex regions in 2-D. Their algorithm requires that obstacles be approximated by iso-oriented rectangles.

Mapping to a Jointed Arm

We provide a mapping to a jointed arm which has an even number of links of the same length. First, group links into pairs of consecutive links. Then, place odd numbered joints $(1, 3, \ldots)$ on the continuous solution in such a way that they are equi-distant. The positions of even numbered joints $(2, 4, \ldots)$ are automatically determined in the process.

Using this every-other-joint mapping, the trajectory for the continuous manipulator in Fig. 6 is mapped to a trajectory for an arm with 12 joints in Fig. 11.

The mapping error is evaluated as follows. Since the every-other-joint mapping is a local mapping scheme, only the mapping for two consecutive links has to be considered. Furthermore, if we assume the following, only two cases, a single arc case and a tangent arcs case, are left in terms of errors (Fig. 12).

Each cubic spiral segment (including the straight line segments at both ends, if they exist) is longer than $2 * l$, where l is the length of each link of the jointed arm.

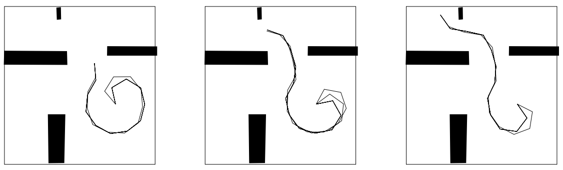

Figure 11: Jointed Arm Trajectory. Only mappings of Frames 2,4,6 of Fig. 6 are shown here.

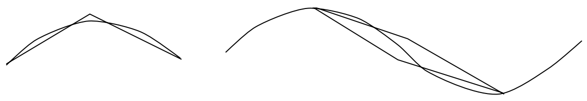

Figure 12: **(Left)** Single arc case: both ends of the link pair are on the same cubic spiral. **(Right)** Tangent arcs case: both ends are on consecutive cubic spirals with opposite sign of curvature. Tangent arcs with the same curvature sign is similar to the single arc case and is less critical.

In order to evaluate the single arc case, we use a circular arc whose curvature is equal to the maximum curvature of the cubic spiral. This gives us an upper bound on the error. In order to evaluate the tangent arcs case, we enumerate pairs of tangent cubic spiral arcs of various turning angles to obtain the error bound. Fig. 13 shows a graph for the errors obtained for the critical tangent arcs cases as a function of α, given the following three maximum curvature constraints.

$$\kappa_{max} = \begin{cases} \frac{1}{2.0l} \\ \frac{1}{1.0l} \\ \frac{1}{0.5l} \end{cases} \tag{4}$$

The relative error is plotted. Each error function decreases in a rage of α where the maximum curvature constraint becomes relevant to the error analysis. The maximum value for the error functions increases with κ_{max}, the maximum curvature constraint. As seen in the graph, the relative error does not exceed 22% for $\kappa_{max} = \frac{1}{1.0l}$.

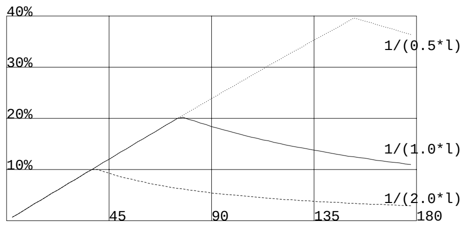

Figure 13: Relative Error for Tangent Arcs Case as Function of α

The tangent arcs case has larger errors than the single arc case, and we summarize the results as follows.

Proposition 2 *Let l be the length of each link. The error of the every-other-joint mapping does not exceed $0.22 * l$, if the following conditions are satisfied.*
(1) Each cubic spiral segment is longer than $2 * l$.
(2) The maximum curvature of cubic spiral segments is below $1/l$.

In fact, the path shown in Fig. 6 was obtained by first growing the obstacles in Fig. 11 by $0.22 * l$ and then planning a path for the continuous manipulator with the above two conditions. The proposition guarantees that the mapping back to a path for the jointed arm will yield a collision free path.

Conclusions

We have presented a path planning method for highly redundant manipulators by means of a continuous model, which captures a macroscopic shape of highly redundant manipulators.

The path planning problem has been shown to be $PSPACE$-complete in terms of DOF of the manipulator [Reif 79, Canny 88]. Our approach overcomes the complexity with a strong heuristic: utilizing redundancy by means of the continuous model. The continuous model allows us to change the complexity of the planning problem from a function of both the DOF of the manipulator (believed to be exponential) and the complexity of the environment (polynomial), to a polynomial function of the complexity of the environment only.

DOF of the manipulator is a resource to be utilized in our approach, because the error bound on the mapping improves with the number of DOF of the manipulator. Essentially, we have transformed the problem of planning paths for highly redundant manipulators to the problem of finding smooth paths for point robots. The smooth path planning problem is a new subject in the field (see [Jacobs and Canny 89] for a related problem), and we expect improvements on the algorithm by using more computational geometry.

References

[Brooks 83a] R. A. Brooks. Solving the find-path problem by good representation of free space. *IEEE transaction on Systems, Man and Cybernetics*, 13:190–197.

[Brooks 83b] R. A. Brooks. Planning collision-free motions for pick-and-place operations. *The International Journal of Robotics Research*, 2(4).

[Canny 88] J. Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the ACM symposium on Theory of Computing.*

[Chirikjian and Burdick 90] G. S. Chirikjian and J. W. Burdick. An obstacle avoidance algorithm for hyper-redundant manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation.*

[Clement and Iñigo 90] W. I. Clement and R. M. Iñigo. Design of a snake-like manipulator. *Robotics and Autonomous Systems*, 6:265–282.

[Drozda 84] T. J. Drozda. The spine robot... the verdict's yet to come. *Manufacturing Engineering*, pages 110–112.

[Hayashi 91] A. Hayashi. Geometrical motion planning for highly redundant manipulators using a continuous model. PhD diss. Dept. of Computer Science, The University of Texas at Austin.

[Hirose *et al.* 83] S. Hirose, T. Kado, and Y. Umetani. Tensor actuated elastic manipulator. In *Proceedings of the Sixth World Congress on Theory of Machines and mechanisms.*

[Ivanescu and Badea 84] M. Ivanescu and I. Badea. Dynamic control for a tentacle manipulator. In *Proceedings of the International Conference on Robotics and Factories of the Future.*

[Jacobs and Canny 89] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation.*

[Kanayama and Hartman 89] Y. Kanayama and B. I. Hartman. Smooth local path planning for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation.*

[Khatib 86] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotic Research*, 5(1):90–98.

[Lozano-Pérez 83a] T. Lozano-Pérez. Robot programming. *Proceedings of IEEE*, 71(7):821–841.

[Lozano-Pérez 83b] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120.

[Martelli 77] A. Martelli. On the complexity of admissible search algorithms. *Artificial Intelligence*, 8:1–13.

[Morecki *et al.* 87] A. Morecki, K. Jaworek, W. Pogorzelski, T. Zielinska, J. Fraczek, and G. Malczyk. Robotics system - elephant trunk type elastic manipulator combined with a quadruped walking machine. In *Proceedings of the Second International Conference on Robotics and Factories of the Future.*

[Pieper 68] D. L. Pieper. *The kinematics of manipulators under computer control.* PhD diss., Mechanical Engineering Dept., Stanford Univ.

[Reif 79] J. H. Reif. Complexity of the generalized movers' problem. In *Proceedings of the 20th IEEE symposium on Foundations of Computer Science (San Juan, Puerto Rico).*

[Rueb and Wong 87] 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.

[Singh and Wagh 87] J. S. Singh and M. D. Wagh. Robot path planning using intersecting convex shapes: Analysis and simulation. *IEEE journal of Robotics and Automation*, RA-3(2):101–108.

[Stoker 69] J. J. Stoker. *Differential Geometry.* Wiley-Interscience.

[Todd 86] D. J. Todd. *Fundamentals of robot technology.* John Wiley and Sons.