In *Proceedings of the 1992 IEEE International Conference on Intelligent Robots and Systems (IROS'92)*

A Continuous Approach to Robot Motion Planning with Many Degrees of Freedom

Akira Hayashi[*]
Faculty of Computer Science and System Engineering
Kyushu Institute of Technology
680-4 Kawazu, Iizuka 820, Japan

Benjamin Kuipers[†]
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712, U.S.A.

Abstract

There is a need for highly redundant manipulators to work in complex, cluttered environments. We explore kinematics and path planning for highly redundant manipulators by means of a *continuous manipulator model*, which captures the macroscopic shape of highly redundant manipulators.

A path for the continuous manipulator model can be found by finding a smooth path for its end-effector under a maximum curvature/torsion constraint. Free space decomposition into primary convex regions is suited to finding such smooth paths.

On top of the free space representation, we have developed an algorithm to find a continuous curvature path in 2-D with a maximum curvature constraint. We report on an experiment conducted to measure the efficiency of the algorithm. The 2-D path planning algorithm could be used for path planning in 3-D space by restricting the manipulator movement. Alternatively, 3-D free space can be decomposed into primary convex regions.

The path planning problem has been shown to be $PSPACE$-complete in terms of DOF of the manipulator. However, 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.

1 Introduction

1.1 Highly Redundant Manipulators

Redundant manipulators have more degrees of freedom (DOF) than necessary to specify a tip position and orientation in the workspace (3 in 2-D, 6 in 3-D). Additional degrees of freedom may be necessary for a manipulator whose task includes avoiding obstacles. 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) [22], elastic manipulator [13], spine robot [7, 27], tentacle manipulator [14], elephant's trunk like elastic manipulator [20], snake-like manipulator [6]. Some were actually built. While many of them are so called continuous arms, highly articulated arms are also studied [12].

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.

1.2 Continuous Manipulator Model

We have been exploring kinematics and path planning for highly redundant manipulators by means of a *continuous manipulator model* [11, 8, 10]. The shape of continuous arms along their 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 essentially a smooth curve with a fixed length. It is controlled by continuously-changing curvature $\kappa(s)$ and torsion $\tau(s)$, intrinsic properties of smooth curves, along the length s of the manipulator.

The continuous model in 2-D is controlled by its curvature. 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, curvature operators are defined to move the points. See Fig. 1.

The continuous model in 3-D is controlled by both curvature and torsion. For each segment, its torsion function

[*]Partially supported by grants from AI Research Promotion Foundation and Nittetsu Kougyou Co.

[†]Supported in part by NSF grants IRI-8905494, IRI-8904454, and IRI-9017047, by NASA grant NAG 9-512, and by the Texas Advanced Research Program under grant no. 003658-175.

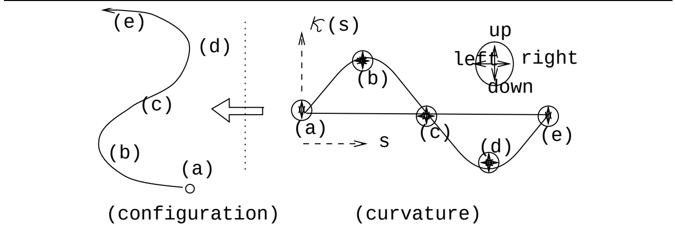

Figure 1: Curvature Segment Representation and its Operators. The following *curvature operators* are used to change curvature (and configuration). **a.** Increase/decrease κ_a, κ_b, κ_c, κ_d, or κ_e. **b.** Increase/decrease s_b, s_c, or s_d. **c.** Rotate the base.

$\tau(s)$ 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 [26]).

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

$\mathbf{v}_1(s)$, $\mathbf{v}_2(s)$, $\mathbf{v}_3(s)$ 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.

[5] 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.

1.3 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 [18].

To plan a path for the continuous manipulator model, it is natural to restrict its motion to follow-the-leader type (snake-like) motion, in which its tip trajectory is traced by its succeeding parts. Even for manipulators with finite number of joints, which cannot precisely trace its end-effector trajectory, it is possible to trace the trajectory within some error bound. We have found the error bound (Δ) can be expressed as a function of the maximum curvature (κ_{max}) of the end-effector trajectory in 2-D space for a class of smooth curves (see Section 4).

Hence, the path planning proceeds as follows:

1. Guess a value of κ_{max}.
2. Grow obstacles by $\Delta(\kappa_{max})$.
3. Attempt to find a smooth, collision-free trajectory from start to goal for a point robot (i.e. end-effector) within maximum curvature κ_{max}.
4. If this fails, revise κ_{max}.
5. If this succeeds, Map the solution to a jointed arm.

We demonstrate that the path planning problem for a highly redundant manipulator can be reduced to the problem of planning a smooth path for a *point robot* within the *same environment* (i.e., not the configuration space), while satisfying a maximum error bound and a maximum curvature constraint.

We have developed an algorithm to find a continuous curvature path within a maximum curvature constraint, based on a decomposition of free space into primary convex regions [24, 25]. The algorithm is efficient in practice. We report on an experiment conducted to measure the performance of the algorithm.

There are two feasible approaches for extending this 2-D path planning approach to 3-D space. [3] 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 [2]. 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 developed for 2-D can be used without modification.

Alternatively, we decompose 3-D free space into primary convex regions [9]. Smooth 3-D curves with curvature and torsion will be used to make turns from one such region to another.

1.4 Organization of the paper

The rest of the paper proceeds as follows. Section 2 explains the algorithm to find a curvature continuous path under a maximum curvature constraint, using primary convex regions. Section 3 discusses the complexity and the actual performance of the algorithm. Mapping back to a jointed arm and its mapping error are discussed in Section 4.

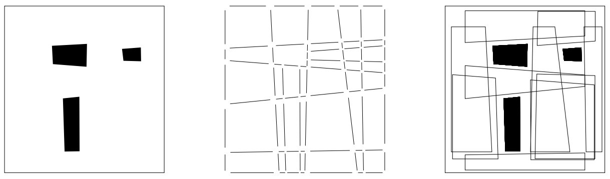
Figure 2: Wall Segments and Primary Convex Regions. Obstacle walls are shown as line segments. Each region is shrunken for visibility.

2 Planning a Smooth path for a Point Robot

In this section, we present an algorithm to find a smooth path (i.e. a continuous curvature path) under a maximum curvature constraint for a point robot.

Smoothness of path is critical for autonomous vehicle navigation and there are algorithms to find a path which consists of straight line and tangent circular arc segments [29, 28, 15, 1]. However, curvature discontinuity exists at every tangent point in these paths, because circular arcs have constant curvature equal to the inverse of their radius and straight lines have zero curvature. [16] have proposed to use cubic spiral curves in order to make a smooth (i.e. continuous curvature) move from one position and orientation to another. Cubic spirals can be constructed to have zero curvature at tangent points. But they did not address the path planning problem.

We find a smooth path which consists of straight lines and cubic spirals with a bound on the maximum curvature of cubic spirals. Our algorithm is based on decomposing free space into primary convex regions. We naturally extend previous algorithms which find straight line paths. Overlapping regions of the primary convex regions are used to make smooth turns from one region to another. Because of the convex nature of free regions, we can adjust the places of turns easily while keeping a path within free space. We find the shortest smooth path using standard graph search techniques for the connectivity graph which is built on top of the representation.

2.1 Free Space Decomposition into Primary Convex Regions

We assume obstacles are given as a set of polygons. We first decompose free space into convex regions.

We need to select a primitive for free space decomposition best suited to our task: finding smooth paths. One of the earliest papers on free space decomposition is [3]. In his algorithm, free space is decomposed into generalized cones which are considered to be free ways. But the robot is supposed to pass along the spines of generalized cones, which will leave less room for our path optimization to make smooth turns.

[17] decompose free space into disjoint polygons: passage regions and channel regions. Passage regions are supposed to correspond to open space (rooms, squares,..), and channel regions to paths connecting open space. Passage regions may be good candidates for smooth turns. However, large open space does not necessarily become a passage region, because passage regions and channel regions are determined only from the topology of obstacle layout without considering the size of areas. Moreover, since they decompose free space into *disjoint* convex regions, they miss straight line path segments if they exist, and the path obtained will have more line segments (and more turns) than other methods.

[24] propose the notion of *primary convex regions.* A primary convex region (PCR) is an unobstructed convex region with each boundary edge covering some portion of an obstacle wall (See Fig. 2). Since each edge of a primary convex region covers some portion of an obstacle wall, the region seems to be natural. In addition, PCR can be seen as an extension of the passage region in [17]. In fact, for any passage region, there exists some primary convex region which contains the passage region. PCRs are suitable representation for our task, because of the following reasons.

- The convexity of PCRs makes the path optimization easier. To move from one PCR to another for a polygonal path, we can make a turn anywhere in their overlap, while staying within the free space.

- PCRs are maximal (in area) convex regions surrounded by obstacle edges. It is easier to find a long straight line path segment, and hence is easier to obtain paths which have less turns.

We have implemented the algorithm in [24] to find PCRs given obstacles. PCRs are found by a directed search for a set of fundamental circuits in an abstract graphical representation of the environment geometry.

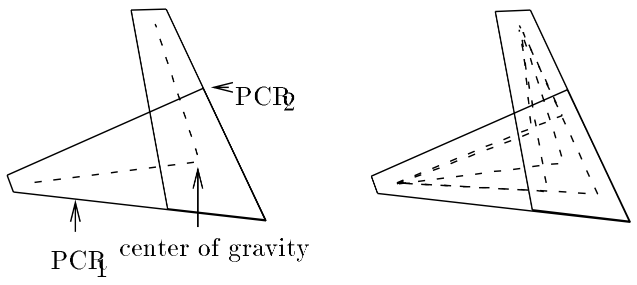

Figure 3: SCTC and MCTC in a OVR

2.2 Candidate Turning Corners

Let us call an overlap of two PCRs an OVR (overlapping region). Finding a polygonal path is easy, once PCRs are obtained. To move from one PCR to another PCR, a turn can be made anywhere within their OVR. However, to make a smooth turn from one PCR to another PCR while satisfying the maximum curvature constraint, and to find a shorter path to reach a goal, we may need to locate *turning corners* appropriately in their OVR. Turning corners are corners of a collision-free polygonal path, which will be made smooth by inserting cubic spiral curves.

We provide two options for locating turning corners. For each OVR,

1. Use its center of gravity as a *single* candidate turning corner (SCTC option, in short).
2. Put *multiple* candidate turning corners (MCTC option, in short). In addition to the center of gravity, candidate turning corners are put between the center of gravity and the vertices of the OVR.

See Fig. 3. Thanks to the convexity of PCRs, the line segments before and after the turn are guaranteed to be within the PCRs.

2.3 Making Smooth Turns 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 the curvature) is described by a cubic function of path distance s.

[16] have developed a method to make a smooth move from one position and orientation to another, using cubic spiral curves.

Proposition 1 (Kanayama and Hartman) *If the size d and the deflection α of a cubic spiral is given (Fig. 4), its length l, curvature κ are*

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

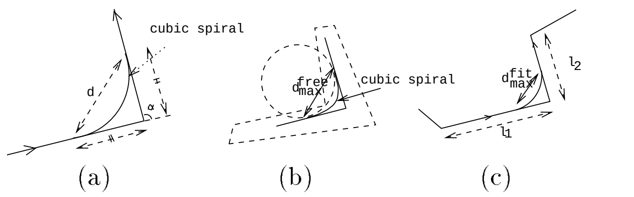

Figure 4: **(a)** Smooth Turn using a Cubic Spiral **(b)** d_{max}^{free} **(c)** d_{max}^{fit}

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

where

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

Their 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_{max}^{free}, the maximum d for which the curve lies entirely within free space.
3. Find d_{max}^{fit}, the maximum d for a cubic spiral to fit along both tangent line segments.
4. Check $d_{min} \leq min(d_{max}^{free}, d_{max}^{fit})$. 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:

$$\kappa_{max} = \kappa(0) = \frac{1.5\alpha D(\alpha)}{d} \tag{4}$$

Therefore,

$$d \geq d_{min} = \frac{1.5\alpha D(\alpha)}{\kappa_{max}} \tag{5}$$

To find d_{max}^{free} is not easy, because cubic spirals are expressed via curvature. However, cubic spirals are 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 largest radius r_{max}, we use the condition that the arc passes through one of the corners of the OVR (Fig. 4). And we get

$$d \leq d_{max}^{free} = 2r_{max}sin(\alpha/2) \tag{6}$$

It is possible to find whether we can fit smooth turns by using d_{min} and d^{free}_{max} obtained, *given a whole candidate polygonal path.* However, this leads to an exhaustive search, because each turn interferes with its preceding and following turns. To avoid an exhaustive search, we use a local fit method. When making a turn, we limit its starting/ending point 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. 4). To make a turn within l_{min}

$$d \leq d^{fit}_{max} = 2l_{min}cos(\alpha/2) \quad (7)$$

2.4 Graph Search for a Smooth Path

We then 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 end point of such a line segment is either a candidate corner inside an OVR with another PCR or the initial or goal position of a point robot. 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 2.3. The cost (length) of the edge is the length of the partial path (a cubic spiral or a line segment)

1. from the start point of L_i to the midpoint of L_j, if the start point of L_i is the initial position,
2. from the mid point of L_i to the end point of L_j, if the end point of L_j is the goal position,
3. from the midpoint of L_i to the midpoint of L_j, elsewhere.

We use the A^* algorithm (see [21]) 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. 5 shows the steps involved in the path planning.

2.5 Finding a Path for Manipulator

We assume there is enough open space around the base to fold the manipulator. To achieve a position, the continuous manipulator is retracted, rotated, and then extended along a path. To find the path, the algorithm for a point robot is extended as follows. 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 OVRs 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 path in Fig. 6.

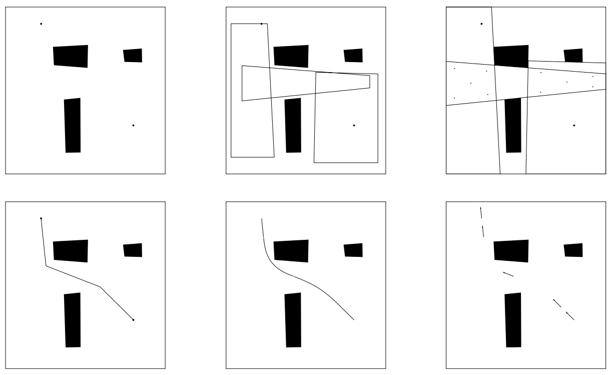
Figure 5: Steps Involved in Path Planning. **(1)** Initial and goal position is given. **(2)** Identify PCRs. (only those on the solution path are shown.) **(3)** Put candidate turning points in OVRs. **(4)** Find least cost path in connectivity graph, consistent with maximum curvature constraint. **(5)**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

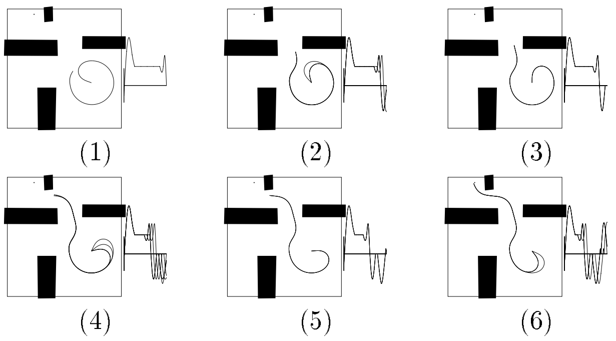

Figure 6: Extending the manipulator along a path

3 Experimental Results

3.1 Complexity

A loose upper bound on the complexity of the algorithm in Section 2 is obtained as follows. An upper bound on the algorithm to find PCRs is $O(n^4)$ time in the number of obstacle edges [24] *(factor 1).* 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 a node in the graph is determined by a sequence of 3 PCRs *(factor 2)* and there cannot be more than $O(n^4)$ PCRs. By using the algorithm in [19] which improves the exponential worst case running time of A^* and runs in the square of the number of nodes *(factor 2)*, we obtain $O(n^{24})$ as an upper bound.

example	obstacle edges	PCRs	candidate corners	nodes in graph	path length	search time (sec)
1	12	10	SCTC	244	701	1
			MCTC	5236	630	12
2	26	24	SCTC	1604	705	4
			MCTC	15044	667	28
3	43	41	SCTC	6302	924	111
			MCTC	80876	875	641
4	101	96	SCTC	31674	963	1044
			MCTC	295672	875	23210

Table 1: Search Time and Path Length with SCTC and MCTC. Search time was measured on a Sun Spark Station 2.

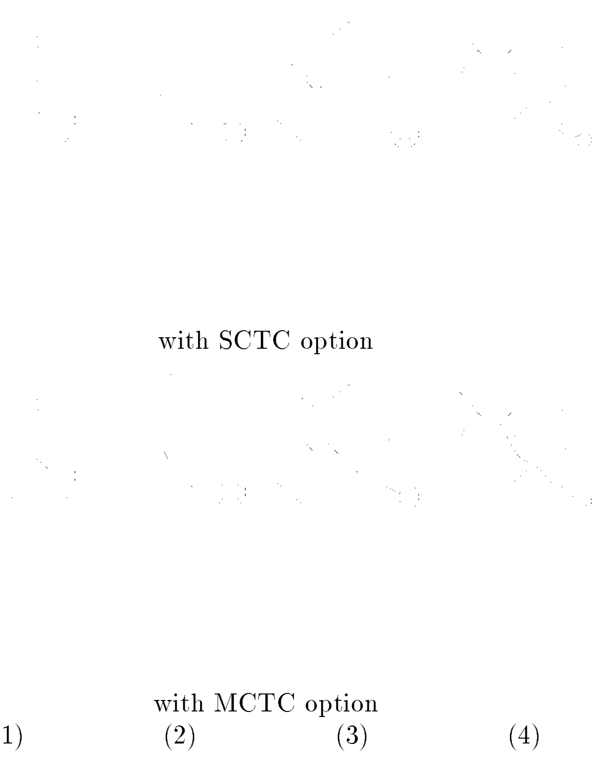

Figure 7: Smooth paths found for 4 layouts of obstacles. Circles in the right bottom show the minimum turning radius given for the search (inner circles), and the one obtained for the path (outer circles).

3.2 Experiment

Although the upper bound obtained is a high order polynomial, the algorithm is efficient in practice. We have conducted an experiment to measure the performance of the algorithm. Fig. 7 are the paths found for the search in four layouts of obstacles. For each of the obstacle layouts, two different conditions for the search are given, on the basis of the number of candidate turning corners in the OVRs: SCTC or MCTC, as explained in Section 2.2. By selecting MCTC option, we can find shorter paths at the cost of longer search time. Table 1 shows the times taken to find the paths in Fig. 7.

We observe the followings from the experiment.

- About *factor 1*: The total number of PCRs is proportional to the number of obstacle edges (see Table 1). This is in accordance with Rueb and Wong's simulation result. They have reported an $O(n)$ performance result for their experiment as opposed to the $O(n^4)$ upper bound. This immediately makes our algorithm run in $O(n^6)$ time instead of $O(n^{24})$.

- About *factor 2*: This cubic factor is obtained by assuming that each PCR overlaps (i.e. intersects) with all other PCRs. This seems to be too pessimistic. The actual factor is quadratic rather than cubic (see Fig. 8).

The actual search time as a function of the number of obstacle edges, n, is $O(n^4) \sim O(n^5)$ in the experiment (see Fig. 9), which may be explained from the above observations on factors 1 and 2.

4 Mapping to a Jointed Arm

We provide a mapping from the continuous manipulator model 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 equidistant. The positions of even numbered joints $(2, 4, \ldots)$ are automatically determined in the process.

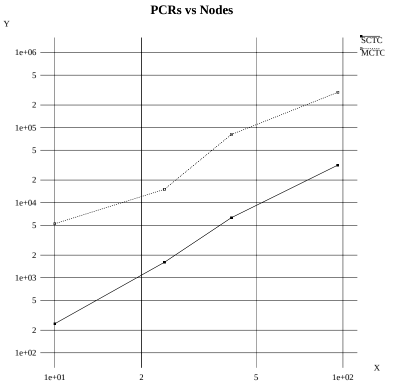

Figure 8: This graph plots, on a log-log scale, the number of nodes in a connectivity graph (Y) as a function of the number of PCRs (X).

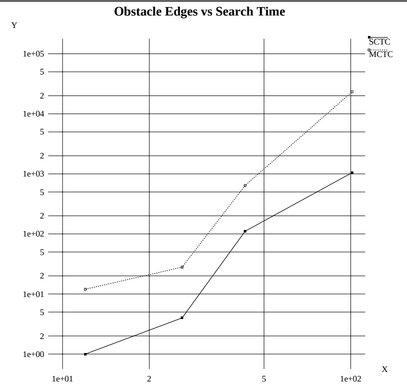

Figure 9: This graph plots, on a log-log scale, the actual search time (Y) as a function of the number of obstacle edges (X).

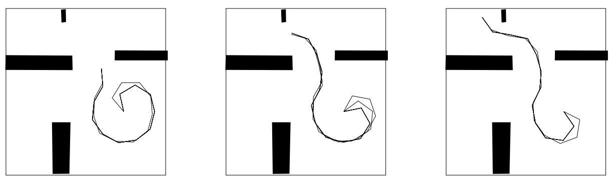

Figure 10: Jointed Arm Trajectory. Only mappings of Frames 2,4,6 of Fig. 6 are shown here.

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. 10.

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. 11).

> 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.

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. 12 shows a graph

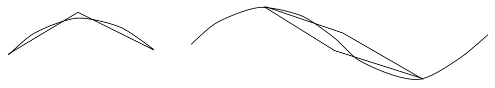

Figure 11: **(Left)** Single arc case: both ends of the link pair are on the same cubic spiral. **(Right)** Tangent arcs case: both ends are on consecutive cubic spirals with opposite sign of curvature. Tangent arcs with the same curvature sign is similar to the single arc case and is less critical.

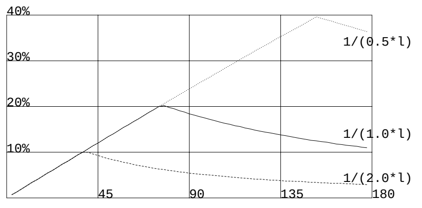

Figure 12: Relative Error for Tangent Arcs Case as Function of α

for the errors obtained for the critical tangent arcs cases as a function of α, given the following three maximum curvature constraints.

$$\kappa_{max} = \begin{cases} \frac{1}{2.0l} \\ \frac{1}{1.0l} \\ \frac{1}{0.5l} \end{cases} \qquad (8)$$

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 $\kappa_{max} = \frac{1}{1.0l}$.

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. 10 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.

5 Summary and Conclusions

Path planning for highly redundant manipulators has been explored by means of the continuous manipulator model, which captures the macroscopic shape of highly redundant manipulators. A path for the continuous manipulator model can be found by finding a smooth path for its end-effector under a maximum curvature/torsion constraint. We have showed that free space decomposition into primary convex regions is suited to finding such smooth paths.

On top of the free space representation, we have developed an algorithm to find a continuous curvature path in

2-D with a maximum curvature constraint. An experiment has been conducted to show that the algorithm is efficient in practice.

The path planning problem has been shown to be $PSPACE$-complete in terms of DOF of the manipulator [23, 4]. However, 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.

References

[1] A. Basu and J. Aloimonos. Approximate constrained motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1990.

[2] R. A. Brooks. Planning collision-free motions for pick-and-place operations. *The International Journal of Robotics Research*, 2(4), 1983.

[3] 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.

[4] J. Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the ACM symposium on Theory of Computing*, 1988.

[5] 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*, 1990.

[6] W. I. Clement and R. M. I. nigo. Design of a snake-like manipulator. *Robotics and Autonomous Systems*, 6:265–282, 1990.

[7] T. J. Drozda. The spine robot... the verdict's yet to come. *Manufacturing Engineering*, pages 110–112, September 1984.

[8] A. Hayashi. *Geometrical Motion Planning for Highly Redundant Manipulators using a Continuous Model*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, May 1991.

[9] A. Hayashi and S. Kodama. Free space decomposition and path planning in 3d. In *The Second Pacific Rim International Conference on Artificial Intelligence (PRICAI'92)*, pages 294–300, September 1992.

[10] A. Hayashi and B. J. Kuipers. Path planning for highly redundant manipulators using a continuous model. In *The Ninth National Conference of Artificial Intelligence (AAAI-91)*, pages 666–672, July 1991.

[11] A. Hayashi, J. Park, and B. J. Kuipers. Toward planning and control of highly redundant manipulators. In *Fifth IEEE International Symposium on Intelligent Control*, pages 683–688, September 1990.

[12] S. Hirose, K. Ikuta, and Y. Umetani. A new design method of servo actuators based on the shape memory effect. In *Proceedings of Fifth Symposium on Robotics, Man, and System*, 1984.

[13] S. Hirose, T. Kado, and Y. Umetani. Tensor actuated elastic manipulator. In *Proceedings of the Sixth World Congress on Theory of Machines and mechanisms*, 1983.

[14] M. Ivanescu and I. Badea. Dynamic control for a tentacle manipulator. In *Proceedings of the International Conference on Robotics and Factories of the Future*, 1984.

[15] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

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

[17] D. T. Kuan, J. C. Zamiska, and R. A. Brooks. Natural decomposition of free space for path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1985.

[18] T. Lozano-Pérez. Robot programming. *Proceedings of IEEE*, 71(7):821–841, July 1983.

[19] A. Martelli. On the complexity of admissible search algorithms. *Artificial Intelligence*, 8:1–13, 1977.

[20] 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*, 1987.

[21] N. J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., 1980.

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

[23] J. H. Reif. Complexity of the generalized movers' problem. In *Proceedings of the 20th IEEE symposium on Foundations of Computer Science*, 1979.

[24] 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.

[25] 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, April 1987.

[26] J. J. Stoker. *Differential Geometry*. Wiley-Interscience, 1969.

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

[28] G. Wilfong. Shortest paths for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[29] G. T. Wilfong. Motion planning for an autonomous vehicle. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1988.