Efficient resolution of hyper-redundancy using splines

Midhun Sreekumar Menon, Gurumoorthy. B and Ashitava Ghosal

Abstract Hyper-redundant systems such as snake robots, flexible manipulators, ropes and strings discretized as rigid links connected by joints can be reasonably assumed to length preserving during their motion. The resolution of the redundancy in such systems have been addressed by several researchers using least squares and other techniques in which the computation effort increases rapidly with the number of links and thus are not amenable to real time motion planning. In this paper, we present a computationally efficient, tractrix based algorithm which appear more 'natural' with motion of links 'dying' down along the length of the hyper-redundant system. The hyper-redundant system is represented by splines and it is shown that an approximate length preserving motion of the hyper-redundant system can be obtained by employing the tractrix based algorithm on the control polygon which generate the spline. The deviation from the actual length is related to the configuration of the control polygon and it is shown that this approach reduces the dimension of the problem space leading to a very efficient resolution scheme. The approach also has the added advantages of better visualization of the motion due to the higher order continuities and capability of localized shape control available in splines.

Key words: Hyper-redundant system, snake robots, length preserving, knot, splines

1 Introduction

One dimensional flexible objects and algorithms for their natural motion are active areas of research owing to the increased application of snake like robots in various areas such as medical robotics and search and rescue. The main issue in such robots is that they are hyper-redundant and there exists infinite number of solution (or configurations) for a desired motion of the end-effector or the head. Various algorithms have been developed for the resolution of the redundancy and for navigation and motion planning of such robots and kinematic chains. One of the well-known approach uses the pseudo-inverse of the manipulator Jacobian matrix [3]. Obtaining

Midhun Sreekumar Menon, Indian Institute of Science, e-mail: midhun.sreekumar@gmail.com
Gurumoorthy. B, Indian Institute of Science, e-mail: bgm@mecheng.iisc.ernet.in
Ashitava Ghosal, Indian Institute of Science, e-mail: asitava@mecheng.iisc.ernet.in

the pseudo inverse explicitly requires $\mathscr{O}(n^3)$ operations where n is the number of joints and is not feasible if n is large. In another well-known approach with complexity $\mathscr{O}(n)$, continuous curves are used to approximate the backbone curve and motion planning is done on the backbone curve [1]. In this approach the length is approximately preserved. Recently an optimization based, length preserving algorithm for motion planning of redundant manipulators has been proposed [2]. Another algorithm by Su et al. [7] uses inverse kinematics and subdivision algorithms complementarily to generate length preserving motion of hyper-redundant manipulators. In both these algorithms, the computation complexity is $\mathscr{O}(n)$ where n is the number of links in the system. Even these efficient algorithms poses limitations for real-time motion visualization and planning of flexible objects such as ropes, strings, hair or snake robots. In such flexible objects, to obtain realistic motion visualization and higher order continuities of slope (C_1) or curvature (C_2), they need to be discretized into a very large number of linear segments/pieces resulting in a large n and an increase in computational costs. In this paper, we propose a novel method of reducing the computational cost using a tractrix based approach together with splines and the method of sub-division. The tractrix based approach provides the 'natural' motion of the flexible object and the use of splines and sub-division yields a significant reduction in dimension of the joint space with a trade-off that the length of the hyper-redundant system is now approximately preserved.

2 Review of tractrix based approach and splines

In this section, we present a short review of the tractrix based approach and of splines for the sake of completeness. More details about splines are available in any geometric modeling textbook such as Piegl and Tiller [4] and more details on tractrix based approach are available in references [5, 6].

2.1 Tractrix based resolution

Consider a single link of length L moving in the plane. If the head P moves parallel to the X axis and the motion of the tail is *along* the link at each instant, then the tail describes the well-known classical curve known as the tractrix. The equation of the tractrix is the solution of the differential equation

$$\frac{dy}{dx} = \frac{-y}{\sqrt{L^2 - y^2}} \tag{1}$$

given as

$$x = L \log \frac{y}{L - \sqrt{L^2 - y^2}} - \sqrt{L^2 - y^2} \tag{2}$$

Some of the main properties of the tractrix are as follows:

- The infinitesimal $dr = \sqrt{dx^2 + dy^2}$ is the local minimum of all possible infinitesimal displacements of the tail and for a motion of the head dp, $dr \leq dp$.
- The tractrix motion of the link can be extended to 3D space in terms of two differential equations of the form

$$\frac{dy}{dx} = \frac{y - y_e}{x - x_e}, \qquad \frac{dz}{dx} = \frac{z - z_e}{x - x_e} \tag{3}$$

where the equations of the path followed by head are $y_e = m_1 x_e$, $z_e = m_2 x_e$ with $m_1 = y_p/x_p$, $m_2 = z_p/x_p$, and (x_p, y_p, z_p) is the destination point of the head.

Instead of numerically integrating the two differential equations, a computationally intensive process, the following algorithm can be used obtain the location of the tail for a given initial positions of head, tail and the destination point of the head $\mathbf{X}_p = (x_p, y_p, z_p)^T$.

Algorithm *TRACTRIX3D*

1 Define the vector $\mathbf{S} = \mathbf{X}_p - \mathbf{X}_h$ where $\mathbf{X}_h$ is the current location of the head.
2 Define the vector $\mathbf{T} = \mathbf{X} - \mathbf{X}_h$ where $\mathbf{X} = (x, y, z)^T$ is the tail of the link lying on the tractrix.
3 Define the new reference coordinate system $\{r\}$ with the X-axis along $\mathbf{S}$. Hence $\hat{\mathbf{X}}_r = \frac{\mathbf{S}}{|\mathbf{S}|}$.
4 Define the Z-axis as $\hat{\mathbf{Z}}_r = \frac{\mathbf{S}\times\mathbf{T}}{|\mathbf{S}\times\mathbf{T}|}$.
5 Define rotation matrix ${}^0_r[\,R\,] = [\hat{\mathbf{X}}_r \quad \hat{\mathbf{Z}}_r \times \hat{\mathbf{X}}_r \quad \hat{\mathbf{Z}}_r]$.
6 The Y-coordinate of the tail (lying on the tractrix) is given by $y = \hat{\mathbf{Y}}_r \cdot \mathbf{T}$ and the parameter p can be obtained as $p = L\,\mathrm{sech}^{-1}(\frac{y}{L}) \pm |\mathbf{S}|$.
7 Using the parametric form of the tractrix with p denoting the parameter, obtain the X and Y coordinate of the point on the tractrix in the reference coordinate system as

$$x_r = \pm|\mathbf{S}| - L\tanh(\frac{p}{L}) \quad y_r = L\,\mathrm{sech}(\frac{p}{L}) \tag{4}$$

8 Once x_r and y_r are known, the point on the tractrix $(x, y, z)^T$ in the global fixed coordinate system $\{0\}$ is given by

$$(x, y, z)^T = \mathbf{X}_h + {}^0_r[\,R\,](x_r, y_r, 0)^T \tag{5}$$

The algorithm *TRACTRIX3D* can be used for resolution of redundancy for any serial hyper-redundant system. Consider a hyper-redundant manipulator with n rigid links $l_1, l_2, ..., l_n$ with joints $j_1, j_2, ..., j_{n-1}$ where j_i is the joint connecting link i and link $i+1$. Consider the last two links l_n and l_{n-1}. The head of the link l_n, denoted by the point j_n, is required to be moved to a new position $j_{n_{\mathrm{new}}}$ given by $(x_p, y_p, z_p)^T$. From the steps in *TRACTRIX3D*, we can obtain the new location of the tail point j_{n-1} denoted by $(x, y, z)^T$ as it follows a tractrix (see equation (5)). The link l_{n-1} is attached to the link l_n and hence the tail of the link l_n can be considered to be the head of the link l_{n-1}. The head of the link l_{n-1} should now be moved from its existing location to $(x, y, z)^T$. The location of the tail of link l_{n-1}, following a tractrix, can again be obtained from the steps given in algorithm *TRACTRIX3D*. Following similar steps, we recursively obtain the motion of the head and tail of all links down to the first link l_1.

We can make the following remarks about the above resolution scheme.

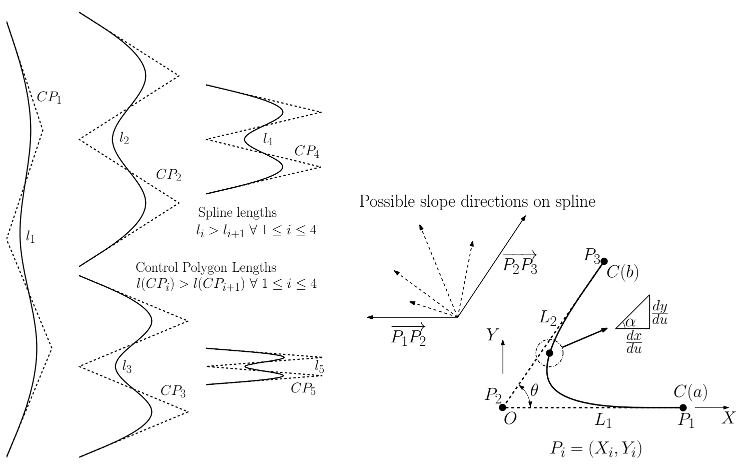
- The algorithm for resolution of redundancy has a complexity of $\mathcal{O}(n)$ where n is the number of rigid links. This fact makes the algorithm amenable for real time computations.
- Under a tractrix motion, when the head of the link l_n moves by dr_n the displacements of all the links obey the inequality $dr_0 \leq dr_1 \leq ... \leq dr_{n-1} \leq dr_n$, with the equality $dr_i = dr_{i-1}$ reached *only* when the line of motion of joint j_i coincides with link l_i. A consequence of this observation is that the motion of the links progressively gets smaller and appears to 'die' out as we move towards the first link. This feature gives a 'natural' looking motion of the hyper-redundant system.

2.2 Representation using splines

Splines are extensively used to represent smooth curves as they possess advantageous properties of local control, and numerical stability. In our approach, as a first step, a control polygon is generated for the smooth spline curve to represent the hyper-redundant system at its initial configuration. It maybe mentioned that the number of legs in the control polygon can be much smaller than the number of links in the hyper-redundant system. Consider a curve as shown in figure 1(a) and the dotted lines which show the control polygon for the curve. As the control polygon is moved using the tractrix algorithm, the curve changes. As shown in figure 1(a), the length of the curve will change even though the length of the control polygon remains constant due to the tractrix based motion. One can intuitively see that as the angle between the edges of the control polygon *decreases* the length of the curve deviates more from the length of the control polygon. This intuitive notion can be mathematically expressed using the equation

$$dl_1^2 = dl_0^2 \left(1 + 2N_3' L_2 du^2 sin(\alpha - \theta) d\theta\right) \tag{6}$$

where u is the parameter for the spline curve, dl_0 is the initial curve length when the included angle is θ, dl_1 is the curve length when the included angle changes by $d\theta$, α is the slope at an arbitrary point on the spline curve, $N_3{}'$ is the derivative of the spline interpolation function for the point P_3, L_2 is the length of the segment P_2P_3. It is well known that the derivative of a B-Spline is again a B-Spline curve. Clearly, as the weighing coefficients are all positive, the slope vector of B-Spline at any point is always a linear combination of $p+1$ adjacent Q_i's which are vectors directed along the control polygon legs with some scaling. In the case of a quadratic curve, slope at any point on the spline will be a convex combination of vectors along two adjacent control polygon segments P_1P_2 and P_2P_3. As shown in figure 1(b), the slope angle is always in the range $\theta \leq \alpha \leq 180$. Hence, the second term in equation (6) is always positive for increasing angles and vice-versa. Thus, as the angle between segments increase, the length of the spline increases and decreases the other way round. Furthermore, when the control polygon is fully stretched out as a straight line, $\theta = 180^\circ$ and $Y_i = 0 \forall\ i$. This can be used to conclude that the length of the curve is always less than or equal to the length of the control polygon.



(a) Spline length and length preserving motion of control polygon

(b) Quadratic spline length dependence on angle

Fig. 1 Spline curve and its control polygon

As shown above, the length of spline changes as the included angles of the control polygon changes. This makes it unusable for redundancy resolution where the length must remain constant. In the next section, we present an algorithm to sub-divide the control polygon when the angle θ becomes too small and the length of the curve changes more than a user defined amount from the original length.

3 Approximate length preservation in splines

From section 2, the spline length and the control polygon length are related by

$$l(CP) = l(C(u)) + E(C(u)) \tag{7}$$

where $E(C(u)) \geq 0$ is length difference at any given stage of the motion. In subdivision algorithm, the control polygon is subdivided by inserting control points. Figure 3 illustrates the idea of subdivision.

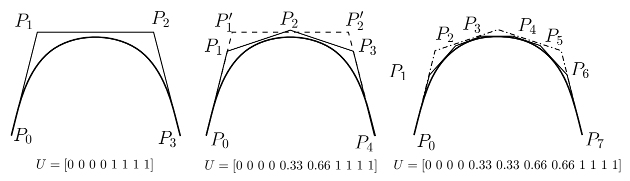

P_1
P_2
P_0
P_3
$U = [0\ 0\ 0\ 0\ 1\ 1\ 1\ 1]$
P_1'
P_2
P_2'
P_1
P_3
P_0
P_4
$U = [0\ 0\ 0\ 0\ 0.33\ 0.66\ 1\ 1\ 1\ 1]$
P_2
P_3
P_4
P_5
P_1
P_6
P_0
P_7
$U = [0\ 0\ 0\ 0\ 0.33\ 0.33\ 0.66\ 0.66\ 1\ 1\ 1\ 1]$

Fig. 2 Subdivision in splines

In the subdivision, P_1P_1' and $P_1'P_2$ is replaced by P_1P_2 in $\triangle P_1P_1'P2$. By triangle inequality, the length of control polygon is reduced. On the other hand, the length of the spline remains the same as before. If original spline and control polygon are denoted by $C_0(u)$, CP_0 respectively and if the spline and control polygon after subdivision is denoted by $C_1(u)$, CP_1 respectively, then we can write from equation (7)

$$l(CP_1) \leq l(CP_0),\ l(C_1(u)) = l(C_0(u)) \Rightarrow E(C_0(u)) \leq E(C_1(u)) \tag{8}$$

After subdivision, the length difference between spline and control polygon decreases. Most importantly, the angles in the introduced edge increases and by equation (6), the sensitivity to change in spline length decreases, thereby effectively reducing error in the spline length. Hence, through subdivision, it is possible to control spline length error by setting a threshold angle value. If the angle between any two segments is less than this threshold ($\theta_i \leq \theta_{th}$), then the control point is subdivided into two new control points. One effect of subdivision is that the number of control points monotonically increases over time depending on the warping of the control polygon and this increases the computation requirement. To overcome this problem, we can also reduce the number of legs in the control polygon when parts of the control polygon stretches out and the included angle crosses a pre-defined threshold. When this is done, the number of legs in the control polygon will reduce and the dimension of the variable space will reduce. The insertion and deletion of links is schematically shown in figure 3.

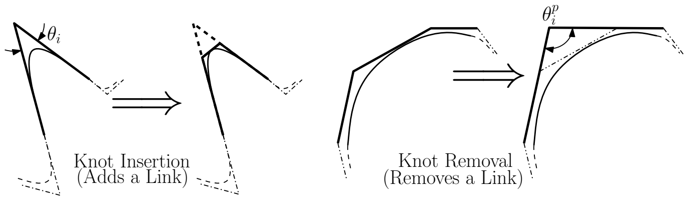


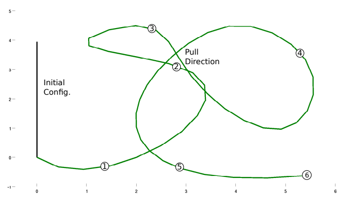
Fig. 3 Knot insertion and knot removal

In the next section, we present numerical simulation results which illustrate the theory presented in the sections above.

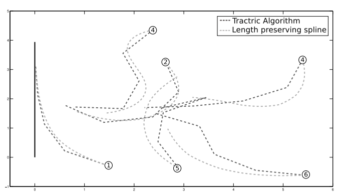
4 Numerical simulation

The tractrix algorithm and spline length preservation algorithm are both simulated on a 4 link planar kinematic chain with path as shown in figure 4(a). The snapshots on various points on the path are shown in figure 4(b). These reveal that one obtains a more natural motion using splines, in addition to savings in computation.

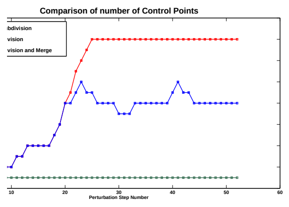
Figure 4(d) shows the variation of control polygon length and figure 4(c) shows number of control points over the simulation duration. As seen, the algorithm adapts to the characteristics of the perturbation path by adding control points as and when required to compensate for the warping of the chain. It also removes control points from the chain as and when the curve can be simplified and represented in terms of a lesser number of control points. In this simulation, threshold for inserting a control point/knot was when the included angle between two segments goes below 140°. Similarly, the threshold for removing a control point/knot was when the two consecutive included angles went above 160°.



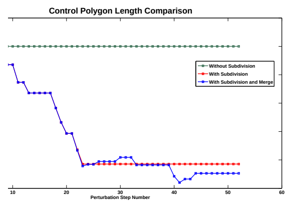
(a) Kinematic simulation in 2D with approximately length preserved spline



(b) Tractrix based simulation



(c) Variation in number of control points



(d) Variation in control polygon length

Fig. 4 Simulation of planar redundant system

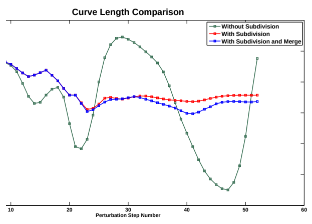


Fig. 5 Length variation of spline during the simulation

Figure 5 shows the length variation of the splines for the algorithms. The variation in length of the spline has been brought down from 1.0 to 0.45, a reduction of 50%, demonstrating that the algorithm works. Moreover, it is seen that the knot removal part of the algorithm is not affecting the length change much (a change of the order of ≈ 0.02), when compared with the gain by subdivision.

Finally, the algorithm is used to simulate a generic curve in 3D space moved along an arbitrary direction. The length of the curve is 30 units and is discretized into 20 links and perturbed in 2650 steps of 0.1, thus making a total motion of 265 units. Figure 6 shows the various snapshots during the motion.

5 Conclusion

This paper proposes a new paradigm for the simulation and visualization of the motion of one-dimensional flexible objects using a tractrix based approach and splines. The tractrix based approach yields a natural motion of the hyper-redundant system and the use of splines leads to efficient computation and more realistic visualization of the motion. An important feature of the proposed algorithm is that it is a

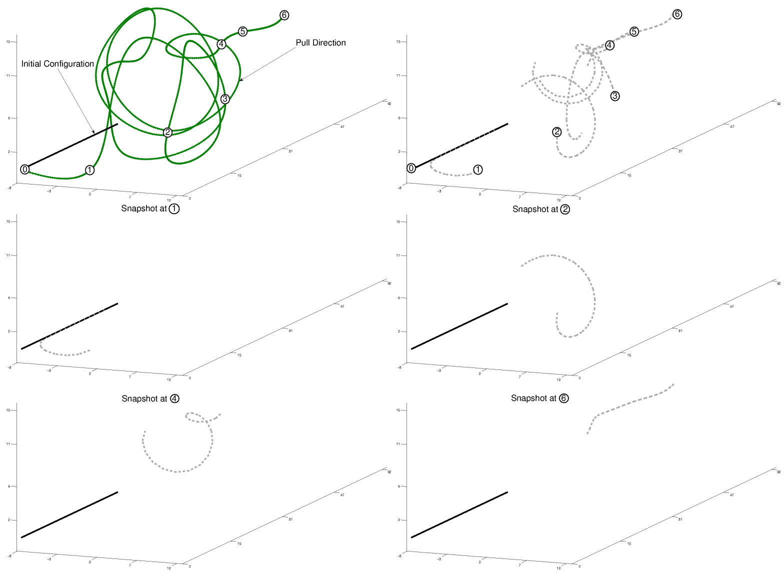


Fig. 6 Motion of an arbitrary curve in a generic direction in 3D

purely *kinematics* and *geometry* based approach. The approach can be applied to simulation and realistic visualization of the motion of generic flexible objects such as snakes, chains, ropes and for redundancy resolution in hyper-redundant robotic manipulators.

Acknowledgment

This work was funded in part by the Robert Bosch Centre for Cyber Physical Systems (RBCCPS) at the Indian Institute of Science, Bangalore.

References

1. Chirikjian, G.S., Burdick, J.W.: A modal approach to hyper-redundant manipulator kinematics. Robotics and Automation, IEEE Transactions on **10**(3), 343–354 (1994)
2. Menon, M.S., Ananthasuresh, G., Ghosal, A.: Natural motion of one-dimensional flexible objects using minimization approaches. Mechanism and Machine Theory **67**, 64–76 (2013)
3. Nakamura, Y.: Advanced robotics: redundancy and optimization. Addison-Wesley Longman Publishing Co., Inc. (1990)
4. Piegl, L., Tiller, W.: The NURBS book (1997)
5. Reznik, D., Lumelsky, V.: Sensor-based motion planning in three dimensions for a highly redundant snake robot. Advanced robotics **9**(3), 255–280 (1994)
6. Sreenivasan, S., Goel, P., Ghosal, A.: A real-time algorithm for simulation of flexible objects and hyper-redundant manipulators. Mechanism and Machine Theory **45**(3), 454–466 (2010)
7. Su, Z., Li, L., Zhou, X.: Arc-length preserving curve deformation based on subdivision. Journal of Computational and Applied Mathematics **195**(1-2), 172–181 (2006). DOI 10.1016/j.cam.2005.03.092