Efficient simulation and rendering of realistic motion of one-dimensional flexible objects

Midhun S. Menon*, B. Gurumoorthy† and Ashitava Ghosal‡
Dept. of Mechanical Engineering
Indian Institute of Science
Bangalore 560 012, India

Abstract

In gross motion of flexible one-dimensional (1D) objects such as cables, ropes, chains, ribbons and hair, the assumption of constant length is realistic and reasonable. The motion of the object also appears more natural if the motion or disturbance given at one end attenuates along the length of the object. In an earlier work, variational calculus was used to derive natural and length-preserving transformation of planar and spatial curves and implemented for flexible 1D objects discretized with a large number of straight segments. This paper proposes a novel idea to reduce computational effort and enable real-time and realistic simulation of the motion of flexible 1D objects. The key idea is to represent the flexible 1D object as a spline and move the underlying control polygon with much smaller number of segments. To preserve the length of the curve to within a prescribed tolerance as the control polygon is moved, the control polygon is adaptively modified by subdivision and merging. New theoretical results relating the length of the curve and the angle between the adjacent segments of the control polygon are derived for quadratic and cubic splines. Depending on the prescribed tolerance on length error, the theoretical results are used to obtain threshold angles for subdivision and merging. Simulation results for arbitrarily chosen planar and spatial curves whose one end is subjected to generic input motions are provided to illustrate the approach.

Keywords: One-dimensional flexible body, Natural motion, Control polygon, Approximate length preservation, Subdivision, Merging

1 Introduction

Motion simulation and rendering of one-dimensional (1D) flexible objects and development of algorithms for real time and realistic or natural motion have been active areas of research in the geometric modeling, CAD and robotics community. The original motivation was the requirement of realism in simulation and display of the motion of cables, ropes, chains, hair, snakes etc. in computer graphics and animation industry. In one of the earlier works, Barzel [1] uses mode shapes of a constrained string model to fake the dynamics of a string. Hergenröether and Däehne [2] discretize the flexible 1D object into a large number of small (linear) rigid objects, each endowed with mass and connected by different kinds of springs and dampers. With appropriate choice of

*Graduate Student. Email: midhun.sreekumar@gmail.com
†Email: bgm@mecheng.iisc.ernet.in
‡Corresponding author. E-mail: asitava@mecheng.iisc.ernet.in, Tel. +91 80 2293 2956

parameter values of mass, spring and damping constant, a physics-based realistic simulation was obtained. Taskıran and Güdükbay [3] have also employed spring-mass system for simulating hair dynamics. In an extension to these works, Güdükbay et al.[4] propose spring mass systems for simulation of elastically deformable models. The main issue in these techniques is to choose or obtain appropriate spring and damping constants for which Natsupakpong and Çavuşoğlu [5] have devised an algorithm for estimating these based on error minimization between FEM and lumped element models. However, the algorithm is not feasible for real-time implementation. Moll and Kavaraki [6] present path planning for flexible 1D objects using minimal energy curves and probabilistic root maps. Ward et al. [7] survey the field of hair modeling, approaching the majority of the existing methods. Grégoire and Schömer [8] and, Spillman and Tescner [9] use Cosserat model for rod-like solids to model bending and torsion for real-time realistic simulation of flexible parts. Lenoir et al.[10] use Lagrangian formulation posing lumped masses on the control polygon vertices and springs along the curve(resisting bending and stretching) combined with spline refinement techniques, namely sub-division/merging to simulate the motion of flexible objects. In an exntension to this work, Theetten et al. [11] generate geometrically exact expressions for deformations of flexible 1D objects by concurrently using beams and spline theory. Goldenthal et al. [12] use a constrained Lagrangian mechanics based approach to handle inextensible cloth simulation – the length constraint (in-extensibility) is explicitly enforced on the cloth mesh thereby increasing computational effort especially when the resolution in the cloth model is increased. To overcome the stiff nature of the differential equation in cloth simulation (due to the high compliance along cloth surface normal vector as compared to almost zero compliance for the in-plane extension), Baraff and Witkin [13] use an implicit solver to simulate cloth motion. This is however an iterative procedure and convergence is an issue. Wang et al. [14] use strain limiting algorithms to overcome the stiffness issue. In another work, Mikchevitch et al. [15] use free-form surfaces and flexible beams to model a real time simulator for assembly-disassembly operations. The review paper by Nealen et al. [16] discuss the existing physically deformable models in computer graphics in detail and gives a very good overview. In all the above mentioned works, dynamics is incorporated and a large amount of effort is towards speeding up the computation or improving the accuracy by adjusting the algorithms. However, all of these methods suffer from one or more issues like stability, convergence, computational errors due to mathematical stiffness of the system, dependence on many arbitrary parameters, phantom forces from high residuals, excessive damping/numerical losses or lack of feasibility for real-time implementation.

In contrast to the above mentioned approaches, many authors have focused on viewing this problem from a pure kinematics perspective so that the issues of stability, convergence and choice of parameters do not arise. Brown et al. [17] have presented tying of knots in a rope with a geometric approach where the flexible 1D object is discretized into linear segments connected by joints and the motion of a trailing segment uses a *follow the leader* based strategy. Su et al. [18] use inverse kinematics and energy minimization to approximately preserve length of deformed polyline and a 4-point subdivision scheme is used to obtain smooth $\mathcal{C}^1$ curve from the deformed polyline. In another work, Sreenivasan et al. [19] use the closed-form equations of the classical *tractrix* curve to iteratively compute the motion of all trailing linear segments. They show that the tractrix based approach has the property of attenuating the motion of the segments from the input end, and this results in a more *natural* motion of the flexible 1D object. In a subsequent work, Menon et al. [20] have shown that the tractrix based solution can be derived from a constrained optimization problem involving minimizing the velocity of points on a curve subject to preservation of the length of the curve. In all of these works, only the kinematics of the 1D object is used to impart realism in the simulation and rendering and since the flexible 1D object is discretized into linear rigid segments, the length is explicitly and always preserved.

In the robotics community, motion planning and simulation of *snake* and other robots with large number of rigid links connected by actuated joints have been a continuing research area (see, for example, ([21, 22, 23, 24]) and the references therein). In a robot, if the number of actuated

joints is more than six for motion in 3D space and more than three for motion in a plane, then there exist many joint angle sets (configurations) which will achieve the same position and orientation of the end-effector of the robot. The main approach in such *redundant robots* is to effectively use the extra actuated joints for selecting poses/paths which optimize a useful functional – this is called the *resolution of redundancy*. One of the earliest techniques used for resolution of redundancy involved the use of the manipulator Jacobian matrix to minimize joint rotation, velocity, torque or to avoid obstacles and singularities in the path of the robot [21]. This approach involves obtaining the pseudo-inverse of the manipulator Jacobian matrix and can have a complexity of $\mathcal{O}(n^3)$ where n is the number of joint variables. Pseudo-inverse based methods are thus not suitable for motion planning when the numbers of links and joints are large. A second approach developed by Chirikjian and Burdick [22] involves the use of a backbone curve to approximate the redundant robot and the motion planning is done on the backbone curve. The complexity of their algorithm is $\mathcal{O}(n)$ but in this approach the length of the curve may not be preserved. In another approach by Reznik and Lumelsky [23], the motion planning is done in the task space (instead of in the joint space) using the classical tractrix curve. As mentioned earlier, the use of the tractrix curve results in a more natural motion of the robot. The tractrix based algorithm has a complexity of $\mathcal{O}(n)$ where n is the number of rigid links.

In recent past, there has been an increased interest in real-time simulation and rendering of the motion of 1D and 2D flexible objects. This is driven by the need to build simulators for laparoscopy, endoscopy and in the general area of training of medical practitioners where motion of blood vessels, tendons etc., motion inside the gastro-intestinal tract or intestine and actions such as tying of knots and suturing needs to be simulated with a high degree of realism [25, 26]. This work has been motivated by the need for developing more realistic simulators for endoscopy and laparoscopic surgeries and is restricted to real-time, efficient and realistic simulation and rendering of the motion of flexible 1D objects.

In this work, the flexible 1D object to be simulated is modeled as a B-spline curve as opposed to being discretized into large number of straight segments. All manipulations are done on the segments of the control polygon which generates the B-spline curve. The main contributions are *a*) obtaining new analytic expressions of change in length of a B-spline curve from an initial configuration as the angle between two adjacent segments of its control polygon is changed, *b*) use of a tractrix based algorithm on the control polygon of a B-spline curve representing the flexible 1D object, and *c*) the development of an adaptive algorithm to approximately preserve the length of the flexible 1D object. The tractrix based algorithm results in a more natural and realistic motion of the curve modeling the flexible 1D object. As the control polygon is moved, the resulting length of the curve is not preserved and the adaptive algorithm is used to sub-divide and merge sides of the control polygon so that a prescribed error tolerance on the length of the curve is maintained at all times during the motion of the curve. Since the angle between the adjacent segments is related to the length of the B-spline curve, monitoring the angle is enough to decide on the subdivision and merging.

Note that the tractrix based algorithm has a complexity of $\mathcal{O}(n)$ where n is the number of segments used to represent the flexible 1D object. However, since the number of sides of the underlying control polygon is much less than n, the complexity of the algorithm can be termed as $\mathcal{O}(1)$. The algorithms for natural and realistic motion planning, the adaptive altering of the control polygon and the mathematical results are illustrated using numerical examples where an arbitrary curve is moved along a generic direction with a prescribed length error tolerance. The efficiency of the developed algorithms are also demonstrated with the numerical examples.

The paper is organized as follows: in Section 2 we briefly present the tractrix based motion of a flexible 1D object. New analytic expressions and results for the length of a quadratic B-spline and cubic B-spline curve in terms of the angle between two adjacent segments of the control polygon are presented. The notion of moving the generating control polygon and resulting change in the length of the spline due to motion of the control polygon is also presented. In Section 3, we present

an algorithm to adaptively subdivide and merge edges of a control polygon to maintain the length of a curve to within a specified length error. In Section 4, we present numerical results illustrating our approach for efficient and realistic motion simulation and visualization of motion of flexible 1D objects. In Section 5 we present the conclusions of this work.

2 Mathematical formulation

In this section, we present the main theoretical results used to obtain the algorithms for efficient and realistic simulation and rendering of the motion of flexible 1D objects. We start with a brief discussion on the tractrix based algorithm and then present new results dealing with length of quadratic and cubic B-splines.

2.1 Tractrix based motion

The tractrix based approach for natural and realistic motion of a curve or a flexible 1D object is dealt in details in references [19, 20]. We present the key results used in this work.

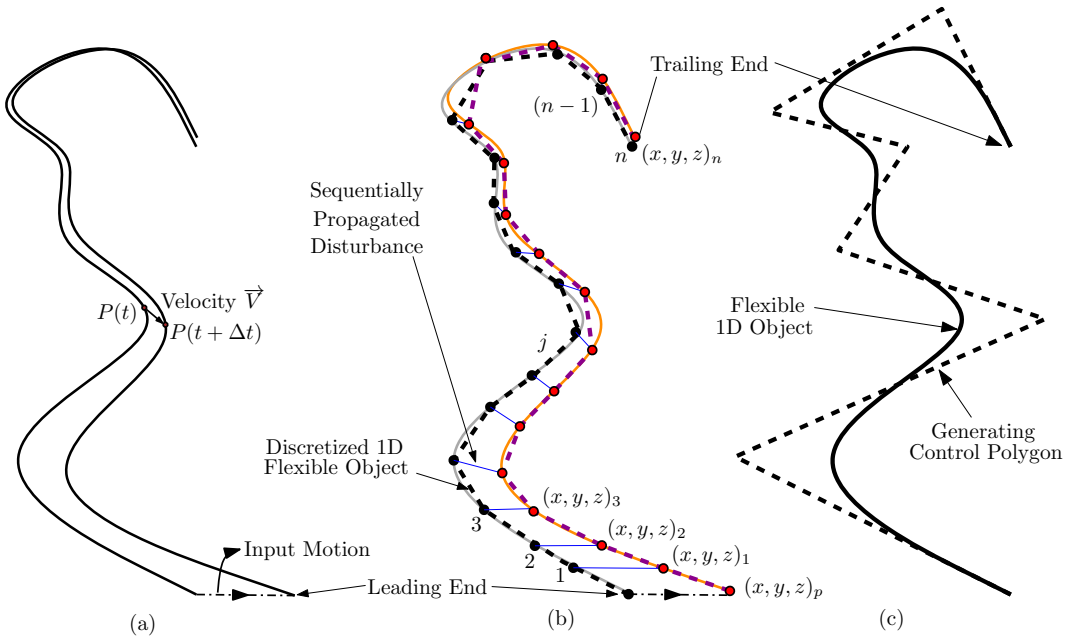


Figure 1: (a)One-dimensional flexible object, (b)Tractrix based motion planning and (c)Generating control polygon

- Consider a curve of length L whose one end is given an input motion as shown in Fig. 1 (a) subject to the constraint that the length of the curve is preserved. As shown, using calculus of variation, the infinitesimal motion at any point on the curve is minimised when the velocity of the point is along the tangent to the curve [20].
- For the special case of a line segment of length L, initially lying along the Y axis and an input motion given to the leading end on the X axis along the X axis, the path (curve) traced by the trailing end of the linear segment is given in a parametric form as

$$x(q) = q - L \tanh\left(\frac{q}{L}\right) \quad \text{and } y(q) = L \operatorname{sech}\left(\frac{q}{L}\right), \tag{1}$$

where q denotes the parameter which in our case is the time t.

The above Eq. (1) is the well-known equation of a classical curve known as the *tractrix* [23]. It can be shown that for a infinitesimal motion, dq, of the leading end, the infinitesimal displacement of the trailing end $dr = \sqrt{dx^2 + dy^2}$ is the minimum of all possible infinitesimal displacements of the trailing end, if it follows a tractrix curve. It can also be shown that $dr \leq dq$. Additionally, as $q \to \infty$ (or $t \to \infty$), $\tanh(q/L) \to 1$ and $\text{sech}(q/L) \to 0$ which gives

$$x(\infty) \to q - L, \qquad y(\infty) \to 0, \tag{2}$$

implying that as time increases, the linear segment aligns with the X-axis and in this case $dr = dq$.

- The tractrix based approach can be applied to *arbitrary 3D motion* of a flexible 1D object discretized by n linear segments as shown in Fig. 1 (b) (see reference [19] for details). This is due to the observation that the destination point $(x, y, z)_p^T$, the leading end or point 1 and the trailing end of the first segment or point 2 *define a plane*. For the arbitrary 3D displacement of the leading end (point 1) to $(x, y, z)_p^T$, the new location of the trailing end (point 2) *in the plane* can be obtained from Eq. (1). The location of the trailing end in 3D space, denoted by $(x, y, z)_2^T$, can then be obtained by using an appropriate rotation matrix. The second segment $\overrightarrow{23}$ and the vector along the displacement of point 2 to $(x, y, z)_2^T$, in general, will lie on a *different plane*. However, again the new location of the trailing end (point 3) given by $(x, y, z)_3^T$ can be computed using the tractrix equations and a different appropriate rotation matrix. Proceeding in a similar manner, the displacement of the leading end of the $(i-1)^{th}$ segment is the displacement of the trailing end of the i^{th} segment and one can thus iteratively go down to the n^{th} linear segment and obtain the new configuration of the discretized flexible 1D flexible object.

- The algorithm described above has a complexity of $\mathcal{O}(n)$ where n is the number of line segments used to discretize the flexible 1D object. Since the line segments are considered as rigid, the sum of the length of the segments is automatically preserved during motion. If n is chosen appropriately to ensure that the curve length is approximately equal to the sum of the length of the line segments, the length of the curve is approximately preserved during the motion.

- When the input end is displaced, the displacements of all the trailing segments obey the inequality $dr_0 \geq dr_1 \geq ... \geq dr_{n-1} \geq dr_n$, with the equality $dr_i = dr_{i-1}$ reached *only* when all the segments align with the input displacement direction. A consequence of this observation is that the motion of the segments progressively gets smaller and appears to 'die' out away from the input end. If the input motion vector is constant in one direction, from Eq. (2) it can be concluded that all the segments eventually align with the direction of the input motion.

The 'dying' out and eventual alignment with the input motion features give the tractrix based approach a more 'natural' and physically realistic motion of the motion of a flexible 1D object. The complexity of $\mathcal{O}(n)$ makes it amenable for efficient simulation and realistic visualization of the motion. The complexity $\mathcal{O}(n)$ can be further reduced if instead of discretizing the flexible object with a large number of rigid linear segments, we approximate the flexible 1D object with a B-spline and we apply the tractrix based motion strategy to the line segments in the control polygon. Fig. 1 (b) shows a flexible 1D object discretized by several line segments and in Fig. 1 (c), the same flexible 1D object is represented by a spline curve, its control polygon and an open uniform knot vector [27] – this ensures that the spline is end point interpolating and ends of the spline match the ends of the flexible 1D object. The number of segments in the control polygon is typically much less than the number of linear segments used to realistically discretize the flexible 1D object– in the illustration the flexible 1D object is discretized by 16 linear segments but the control polygon shown in Fig. 1 (c) has only 7 segments.

2.2 Splines and control polygon

As mentioned earlier, one of the key ideas of this paper is to move the segments of the control polygon instead of the elements of the discretized curve (polyline) to reduce computation and enable real-time simulation and visualization. However, it is well-known that as the control polygon changes, the spline curve and its length changes [27]. This is illustrated in Fig. 2: the left most spline curve of length L_{C_1} is generated by the control polygon CP_1 and as the sides of the control polygon is moved to CP_2 keeping $L_{CP_1} = L_{CP_2}$, one can clearly see that the length of the spline curve changes. It can be observed from Fig. 2 that as the angle between the adjacent segments

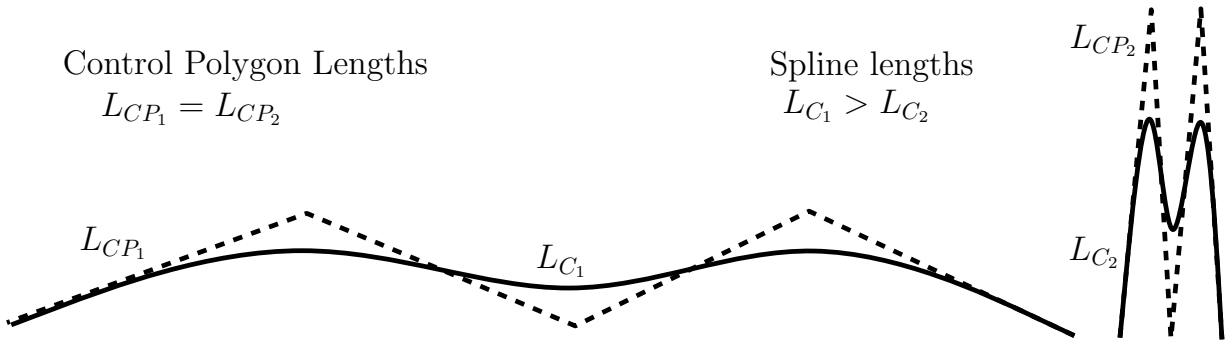


Figure 2: Spline length with length preserving transformations of control polygon

decrease the length of the spline decreases and the upper bound of curve length is the length of the control polygon. We provide mathematical proofs of these two observations next, first for a quadratic B-spline and then for a cubic B-spline. Finally, arguments for any higher degree 3D splines are provided.

2.2.1 Quadratic spline

For a quadratic spline shown in Fig. 3, the three consecutive points P_1, P_2 and P_3 always lie on a plane. It may be noted that the analysis is not restricted to planar quadratic splines as the next three points can lie on a *different* plane and the entire curve can be spatial. Without loss of generality, the coordinates of the three points can be assumed to be $[L_1, 0]^T$, $[0, 0]^T$ and $[L_2 \cos\theta, L_2 \sin\theta]^T$, respectively, where L_1, L_2 are the lengths of the two sides of the control polygon and θ is the angle between the adjacent sides of the control polygon. The set of control points (P_1, P_2, P_3) generate the part of the spline shown in Fig. 3 for the parameter interval $u \in (u_i, u_{i+1})$. The length of the spline curve for $u \in (u_i, u_{i+1})$, is given by

$$l(\theta) = \int_{u_i}^{u_{i+1}} \left(\left(\sum_{i=1}^{3} \frac{dN_{i,p}(u)}{du} X_i \right)^2 + \left(\sum_{i=1}^{3} \frac{dN_{i,p}(u)}{du} Y_i \right)^2 \right)^{\frac{1}{2}} du, \tag{3}$$

where $l(\theta)$ means that the spline curve length depends on the included angle θ.

In an open-uniform knot vector of the form $[u_1\ u_1\ u_1\ u_2\ u_3 \ldots u_{m-1}\ u_m\ u_m\ u_m]$ with $u_1 \leq u_2 \leq u_3 \leq \cdots \leq u_m$, without loss of generality, an intermediate knot interval $(u_i, u_{i+1})(i \neq 1, 2, m-2, m-1)$ can be reduced to $(0, 1)$ by appropriate scaling and translation of parameter u. The basis functions $N_{i,p}$ for a quadratic spline with $p = 2$ and for the knot interval $(0, 1)$ are given by

$$N_{1,2} = \frac{1}{2}(1-u)^2,\ N_{2,2} = -u^2 + u + \frac{1}{2} \text{ and } N_{3,2} = \frac{1}{2}u^2. \tag{4}$$

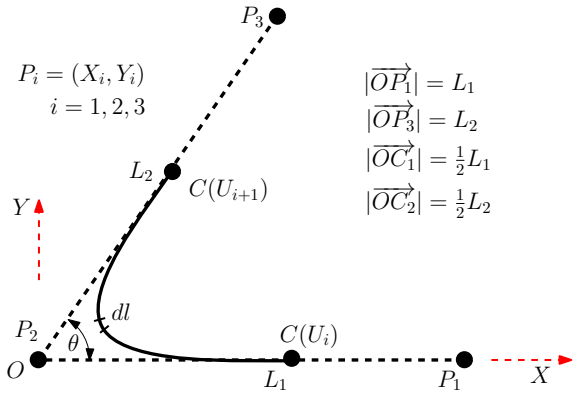


Figure 3: Two segments of the control polygon of a quadratic spline

Using the above, $l(\theta)$ can be simplified to

$$l(\theta) = \int_0^1 \sqrt{(-L_1(1-u) + L_2 u\cos\theta)^2 + (L_2 u \sin\theta)^2}\, du \tag{5}$$

and for $L_1 = L_2 = L$, $l(\theta)$ is given by

$$l(\theta) = \frac{1}{\sqrt{32L^2(1+\cos\theta)}}\left(\sqrt{8L^4(1+\cos\theta)} + L^2(-1+\cos\theta)\log\frac{\sqrt{2}-\sqrt{1+\cos\theta}}{\sqrt{2}+\sqrt{1+\cos\theta}}\right),\ 0 < \theta < \pi. \tag{6}$$

From above, $\lim_{\theta\to\pi} l(\theta) = L$ and this agrees with known result for a quadratic curve (see pp. 82 in [27]). From Eq. (6), $(L - l(\theta))$ is maximum when $\theta \to 0$ and the maximum difference is 50% (see Fig. 5). The expression for $l(\theta)$ when $L_1 \neq L_2$ is more complicated and is given in Appendix A.

In general, for a quadratic spline with n segments in the control polygon, the total curve length can be computed as

$$L_C = \left(\frac{1}{2}(L_s + L_e) + \sum_{i=1}^{n-1} l_i(\theta_i)\right), \tag{7}$$

where L_s, L_e denotes length of starting and ending segment of the control polygon, respectively, L_i denotes the length of the i^{th} control polygon segment and $l_i(\theta_i)$ denotes the length of the portion of the curve defined by the i^{th}, $(i+1)^{\text{th}}$ and $(i+2)^{\text{th}}$ control points.

2.2.2 Cubic splines

In the case of the quadratic spline, the three points generating the spline define a plane and locally the spline is planar in a knot interval. This is not valid for a cubic spline since the four generating points and the resulting cubic spline need not be planar. In Appendix B, we show that the worst case, in terms of difference between the curve length and the length of the control polygon, is obtained when all the four points lie on a plane. Hence, we consider the case of a planar cubic spline and obtain expressions for bounds on this difference.

Fig. 4 shows four interior control points P_i, $i = 1, ..., 4$ on a plane and the two included angle θ_1, θ_2 between the first and second, second and third segments, respectively. The initial configuration of the knot vector is assumed to be a clamped open-uniform knot vector of the form

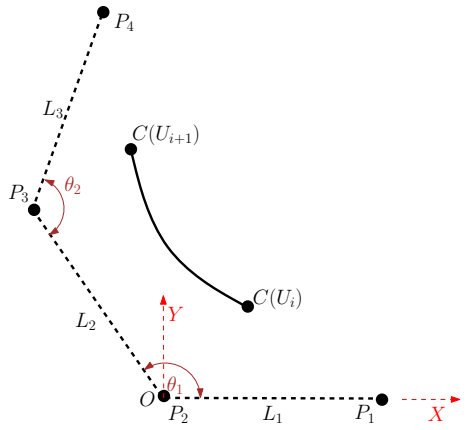


Figure 4: Control polygon for a planar cubic spline

$[u_1\ u_1\ u_1\ u_1\ u_2\ u_3 \ldots u_{m-1}\ u_m\ u_m\ u_m\ u_m]$, $u_1 \leq u_2 \leq \cdots \leq u_m$. Note that for a uniform non-repeated knot vector, the difference between the curve length and control polygon length is maximum as any repetition of knot vector or subdivision pulls the curve towards the control polygon thereby reducing the length difference.

From the figure, the points of the spline are $P_1 = [L_1, 0]^T$, $P_2 = [0, 0]^T$, $P_3 = [L_2 \cos\theta_1, L_2 \sin\theta_1]^T$ and $P_4 = [L_2 \cos\theta_1 - L_3 \cos(\theta_1 + \theta_2), L_2 \sin\theta_1 - L_3 \sin(\theta_1 + \theta_2)]^T$. The elemental length of the cubic spline $C_0(u)$ is given by

$$dl = \left(\left(\sum_{i=1}^{n} \frac{dN_{i,p}(u)}{du} X_i \right)^2 + \left(\sum_{i=1}^{n} \frac{dN_{i,p}(u)}{du} Y_i \right)^2 \right)^{\frac{1}{2}} du, \tag{8}$$

where (X_i, Y_i) are the coordinates of P_i, $i = 1, ..., 4$.

Substituting the X and Y coordinates of the points on the control polygon and using N_i' to denote $\dfrac{dN_{i,3}(u)}{du}$, we get

$$\begin{aligned} dl &= \sqrt{A + B}\ du, \\ A &= \left(\frac{\partial X}{\partial u} \right)^2 = \left(N_1' L_1 + N_3' L_2 \cos\theta_1 + N_4' \left(L_2 \cos\theta_1 - L_3 \cos\left(\theta_1 + \theta_2\right) \right) \right)^2 \text{ and} \\ B &= \left(\frac{\partial Y}{\partial u} \right)^2 = \left(N_3' L_2 \sin\theta_1 + N_4' \left(L_2 \sin\theta_1 - L_3 \sin\left(\theta_1 + \theta_2\right) \right) \right)^2. \end{aligned} \tag{9}$$

For a cubic spline, the basis functions in $u \in [0\ 1]$ are

$$\begin{aligned} &N_{1,3} = \frac{1}{6}(1-u)^3,\ N_{2,3} = \frac{2}{3} + \frac{1}{2}u^3 - u^2, \\ &N_{3,3} = \frac{1}{6} + \frac{1}{2}u - \frac{1}{2}u^3 + \frac{1}{2}u^2 \text{ and } N_{4,3} = \frac{1}{6}(1-u)^3. \end{aligned} \tag{10}$$

Substituting the above in Eq. (9), we get

$$\begin{aligned} dl &= \sqrt{Pu^4 + Qu^3 + Ru^2 + Su + T}\; du, \\ P &= \frac{1}{4}L_1^2 + \frac{1}{4}L_3^2 + L_2^2 + L_1L_2\cos\theta_1 + \frac{1}{2}L_1L_3\cos(\theta_1+\theta_2) + L_2L_3\cos\theta_2, \\ Q &= -L_2L_3\cos\theta_2 - 3L_1L_2\cos\theta_1 - L_1L_3\cos(\theta_1+\theta_2) - L_1^2 - 2L_2^2, \\ R &= \frac{3}{2}L_1^2 + \frac{5}{2}L_1L_2\cos\theta_1 + \frac{1}{2}L_1L_3\cos(\theta_1+\theta_2) - \frac{1}{2}L_2L_3\cos\theta_2, \\ S &= L_2^2 - L_1^2,\; T = \frac{1}{4}L_1^2 + \frac{1}{4}L_2^2 - \frac{1}{2}L_1L_2\cos\theta_1, \end{aligned} \tag{11}$$

and the length of the spline generated in the knot interval [0 1] can be obtained as the integral of the right-hand side in Eq. (11). Unlike in the quadratic case, for a cubic spline, the analytical form of the integral as a function of θ_1 and θ_2 is not known and it is not possible to find analytical expressions for the difference between the total length of the control polygon and B-spline curve length. The length difference is a surface (being a function of θ_1 and θ_2) and can always be found using numerical integration. We can, however, obtain useful approximations to the integral by considering the following:

- We consider the effect of change of one angle at a time, i.e., θ_1 is varied with θ_2 held constant. This is reasonable since during the motion of the control polygon, the angles between the adjacent segments are monitored. When the difference in length between spline and control polygon due to a single angle change becomes large, subdivision is used to reduce the difference (see Section 3) and hence effect of change in one included angle can be considered.
- We consider the case of a control polygon with equal lengths, i.e., $L_1 = L_2 = L_3 = L$.

With the above two assumptions, the integral simplifies to

$$l(\theta_1) = \frac{L}{\sqrt{2}}\int_0^1 \sqrt{1 + (pu^4 + qu^3 + ru^2 - \cos\theta_1)}\; du, \tag{12}$$

$$\begin{aligned} \text{where}\quad p &= \cos(\theta_1+\theta_2) + 2\cos\theta_1 + 2\cos\theta_2 + 3, \\ q &= -2\cos(\theta_1+\theta_2) - 6\cos\theta_1 - 2\cos\theta_2 - 6 \text{ and} \\ r &= \cos(\theta_1+\theta_2) + 5\cos\theta_1 - \cos\theta_2 + 3. \end{aligned}$$

Since $0 \le u \le 1$, the term $|W| = |pu^4 + qu^3 + ru^2 - \cos\theta_1| \le 1$. Hence, we can approximate $(1+W)^{\frac{1}{2}}$ as $\left(1 + \frac{1}{2}W\right)$ by keeping only the first term in the binomial expansion. The first-order length difference between the control polygon and the curve, in the knot interval [0 1] due to a change in θ_1 (with θ_2 held constant) is given by

$$e_{\text{first-order}}(\theta) = \frac{L}{2\sqrt{2}}\int_\theta^\pi \int_0^1 \frac{\partial W}{\partial\theta_1} d\theta_1 du = \frac{1}{120}L\sqrt{2}\left(13\cos\theta - \cos\theta_2 - \cos(\theta+\theta_2) + 13\right), \tag{13}$$

where $e_{\text{first-order}}(\theta)$ denotes the length difference when $\sqrt{1+W}$ is approximated by $1 + (1/2)W$. The plot of the exact length difference obtained by numerically integrating right-hand side of Eq. (11) and the plot of the first-order approximation is shown in Fig. 5. For comparison, the length difference obtained for a quadratic spline is also shown in the figure.

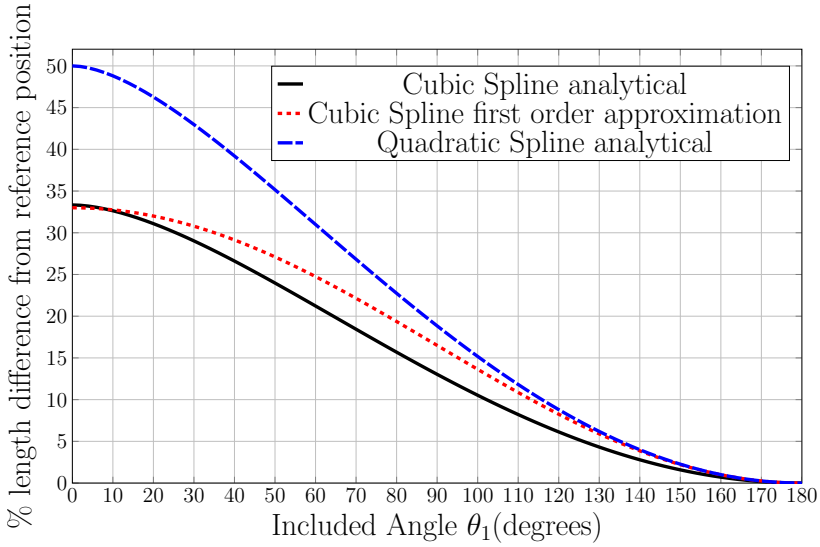


Figure 5: Plot of bound on length difference for a quadratic and cubic B-spline

2.2.3 Summary

We can summarize the results obtained from the quadratic and cubic B-splines as follows:

- For a quadratic B-spline, we can obtain closed-form analytic expression for the curve length as a function of the angle between the adjacent segments. For any angle θ the change in curve length from the completely stretched out case ($\theta = \pi$) can be obtained using Eq. (A-1) as

$$e_{\text{quadratic B-spline}} = l(\pi) - l(\theta) = \left(\frac{L_1 + L_2}{2} - l(\theta) \right). \tag{14}$$

- The difference between the length of the control polygon, $L_{CP} = \sum_{i=1}^{n} L_i$, and the length of the curve L_C is given by

$$E_C = L_{CP} - L_C = \sum_{i=1}^{n} L_i - \left(\frac{1}{2}(L_s + L_e) + \sum_{i=1}^{n-1} l_i(\theta_i) \right). \tag{15}$$

- For a cubic B-spline, there is no known closed-form analytic expression for the curve length as a function of the two angles between the three consecutive segments. The length of the curve can be obtained by integrating Eq. (11) and a numerical plot of change in length with respect to one or both angles can be obtained.

- A first-order approximation of the difference between the length of cubic spline and the length of the generating control polygon, with one angle held constant and equal control polygon segment lengths, can be obtained as shown in Eq. (13). The first-order approximation is conservative to within a maximum of 3.73% difference from the exact length difference obtained from integration.

- It can be seen by comparing the plots in Fig. 5 that a cubic spline gives less length difference when the angle between two adjacent segments change – for a θ value of 100° the length difference for a quadratic spline is 15% whereas for a cubic spline it is 10%. A consequence of this result is that less number of subdivisions (see Section 3) may be required during the motion of the flexible 1D object when it is modeled with cubic splines.
- Although the plots in Fig. 5 are for equal control polygon segment lengths, they serve as a design tool whereby the user can choose a threshold angle to limit the change in curve length within desirable limits.
- When a curve is modeled using cubic splines, non-trivial continuity is guaranteed till third order which implies curvature continuity. Majority of real-world problems demand a maximum of curvature continuity or lesser and hence, the results have been obtained only for a quadratic and cubic spline. However, first-order results along similar lines as the cubic spline can also be developed for higher-order splines.

One key consequence of the above results is that as the control polygon is moved the angle between adjacent segments will change and the length of the spline curve will change. The key idea of length preserving motion of the flexible 1D object will not be possible. In the next Section, we present an adaptive algorithm to approximately preserve the length of the spline curve to within a user specified tolerance when the control polygon is moved.

3 Approximate length preservation in splines

As discussed in Section 2.2, error in length of the B-spline curve from an initial value can be related to the included angle θ between two adjacent segments. The key idea in approximate length preservation is to sub-divide a segment of the control polygon when the included angle between two adjacent segments is less than a threshold value and merge the two adjacent segments when the included angle is larger than another threshold value. In the subdivision step, the control

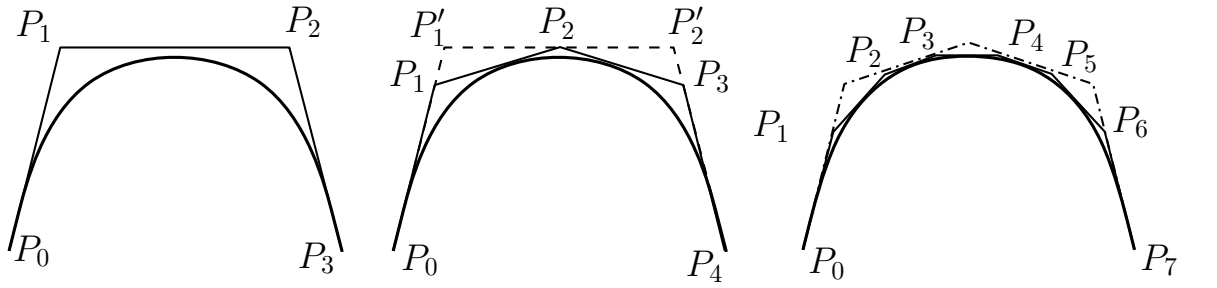


Figure 6: Subdivision in splines

polygon is modified by inserting control points as illustrated in Fig. 6. As shown, the segment P_1P_2 in Fig. 6 (a) is replaced by two segments P_1P_2 and P_2P_3 in Fig. 6 (b) with the original segment P_1P_2 shown as $P'_1P'_2$. From the construction and using triangle inequality in Fig. 6 (b), $|P_1P'_1| + |P'_1P_2| \leq |P_1P_2|$. Likewise $|P_2P'_2| + |P'_2P_3| \leq |P_2P_3|$. Hence the length of control polygon $P_0P_1P_2P_3P_4$ after subdivision is less than the length of the original control polygon $P_0P_1P_2P_3$. If the segments P_1P_2 and P_2P_3 are further subdivided, as in Fig. 6 (c), the total length of the modified

control polygon will decrease even more. In the limit of infinite number of subdivisions, the length of control polygon will coincide with the length of the curve.

One effect of subdivision is that the number of control points (and the number of segments) monotonically increases over time depending on the extent of bending/warping of the control polygon during the motion and this increases the computation requirement in the tractrix based motion algorithm. To overcome this problem, we reduce the number of segments in the control polygon when parts of the control polygon stretches out and the included angle is larger than a pre-defined threshold. The reduction in the number of segments is schematically opposite of subdivision – the sequence for merging is from right to left in Fig. 6. Note that during merging the length of the resulting control polygon increases.

We discuss in detail and present mathematical results for subdivision and merging in the rest of this Section.

3.1 Subdivision in splines

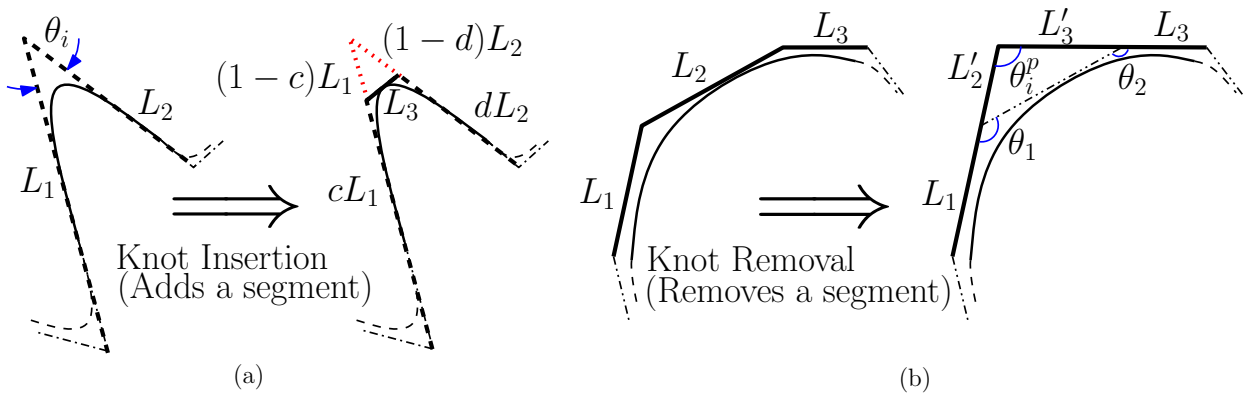


Figure 7: Knot insertion and knot removal

Fig. 7 (a) shows subdivision (also called knot insertion). Assuming the control polygon segments are of lengths L_1, L_2 and the included angle is θ_i, the length of the portion of control polygon before subdivision (L_{CP_0}) and the length after subdivision (L_{CP_1}) is given by

$$L_{CP_0} = L_1 + L_2 \text{ and } L_{CP_1} = cL_1 + dL_2 + L_3 \ (0 < (c, d) < 1), \tag{16}$$

where by using law of cosines

$$L_3 = \sqrt{((1-c)L_1)^2 + ((1-d)L_2)^2 - 2(1-c)(1-d)L_1L_2\cos\theta_i}\ ,$$

and c, d are the ratios for subdivision which can be chosen by user. Hence, the decrease in length of the control polygon after subdivision is given by

$$\begin{aligned}\Delta L_{CP} &= L_{CP_0} - L_{CP_1} \\ &= (1-c)L_1 + (1-d)L_2 - \sqrt{((1-c)L_1)^2 + ((1-d)L_2)^2 - 2(1-c)(1-d)L_1L_2\cos\theta_i}\ .\end{aligned} \tag{17}$$

From the above, as the portion of the control polygon stretches out and the included angle $\theta_i \to \pi$, the change in length of the control polygon $\lim_{\theta_i \to \pi} \Delta L_{CP} \to 0$.

During subdivision, the length of the spline curve remains the *same*. As mentioned earlier, the length of the control polygon, L_{CP}, is more than the length of the spline curve, L_C, and we can write

$$L_{CP} = L_C + e, \tag{18}$$

where $e \geq 0$ is the difference in length between the control polygon and the spline curve. During subdivision, the length of the spline curve is not changed, and from Eqs. (17) and (18), we get

$$L_{CP_1} \leq L_{CP_0} \text{ and } e_1 \leq e_0, \tag{19}$$

where e_0 and e_1 are the length differences before and after subdivision, respectively.

The above proves that the length difference between spline curve and control polygon decreases during a subdivision. Most importantly, the angles between the adjacent segments of the control polygon increases, and, as shown in Fig. 5, increasing the angle reduces the difference in the length between the spline curve and the control polygon. Hence, through subdivision, it is possible to control spline length by setting a threshold angle value θ_{th}. If the angle between any two segments is less than this threshold ($\theta_i \leq \theta_{th}$), then the control polygon is subdivided to obtain two new control points. As θ_{th} increases, length difference on any elemental segment reduces which in turn reduces the total difference in length. Hence the threshold angle plays a key role in the total difference between length of the control polygon and curve. From the analytical results in the previous Section and from Fig. 5, a suitable θ_{th} can be chosen to satisfy a desired length error requirement. In extensive simulation (see Section 4), it is observed that a threshold angle of 140° is found to give total difference less than 5%.

3.2 Merging in splines

When two adjacent segments of a control polygon are merged (by knot removal), the number of segments in the control polygon will reduce and the computations required for the tractrix based motion algorithm will reduce. Knot removal is shown schematically in Fig. 7 (b). As mentioned earlier, in case of knot removal, the length of the control polygon increases and the length before and after knot removal can be written as

$$L_{CP_0} = L_1 + L_2 + L_3 \text{ and } L_{CP_1} = L_1 + L_2' + L_3' + L_3. \tag{20}$$

But we know that $\theta_i^p = (\theta_1 + \theta_2) - \pi$ and by using the law of sines,

$$L_2' = -\frac{\sin\theta_2}{\sin(\theta_1+\theta_2)} L_2 \text{ and } L_3' = -\frac{\sin\theta_1}{\sin(\theta_1+\theta_2)} L_2. \tag{21}$$

From Eq. (20) and Eq. (21), the increase in length of the control polygon can be computed as

$$\Delta L_{CP} = L_{CP_1} - L_{CP_0} = -\frac{\sin\theta_2}{\sin(\theta_1+\theta_2)} L_2 - \frac{\sin\theta_1}{\sin(\theta_1+\theta_2)} L_2 - L_2. \tag{22}$$

Unlike subdivision, the increase in the length of the control polygon is a function of two angles θ_1 and θ_2 and thus lies on a surface. When $(\theta_1, \theta_2) \to \pi$, $\lim_{(\theta_1,\theta_2)\to\pi} \Delta L_{CP} \to 0$ and we can infer that knot removal should be done when the θ_1 and θ_2 are close to π.

Knot removal has the additional complexity of not yielding a unique solution/curve (see pp. 179 in [27] for further details). As shown in Fig. 8, there are two possible solutions (red and blue) for the merged spline curves, which can be derived from the original spline (black). The length of the two spline curves can be computed by integration, and are denoted by L_{C_1} and L_{C_2}. We denote

the control polygon and curve length before merging by L_{CP} and L_C, respectively. Since merging increases the length of the control polygon, we can write

$$L_{CP_1} \geq L_{CP},\ L_{CP_2} \geq L_{CP},\ L_{CP_1} = L_{C_1} + e_1,\ L_{CP_2} = L_{C_2} + e_2, \tag{23}$$

where the two control lengths are denoted by L_{CP_1}, L_{CP_2}, and the difference between the curve length and the control polygon length are denoted by e_1 and e_2. As the length of the two control polygons is known, the terms e_1 and e_2 can be computed and the solution with lower difference between the spline and control polygon length is chosen as the new curve definition. For a knot of

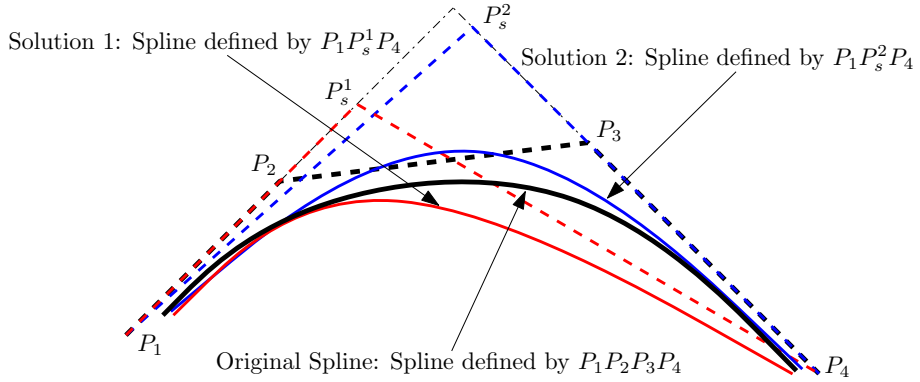


Figure 8: Illustration of multiple solutions encountered in merging process

multiplicity 2 to be removed once, the condition to be satisfied is that two points given below

$$P_s^1 = \frac{P_2 - (1-\alpha_1)P_1}{\alpha_1} \text{ and } P_s^2 = \frac{P_3 - \alpha_2 P_4}{1-\alpha_2}, \tag{24}$$

with $\alpha_1 = \dfrac{u_3 - u_1}{u_4 - u_1}$ and $\alpha_2 = \dfrac{u_3 - u_2}{u_5 - u_2}$ must coincide [27]. As shown in Fig. 8, the two points-P_s^1 and P_s^2, lie on the vector directed along $\overrightarrow{P_1P_2}$ and $\overrightarrow{P_4P_3}$. Substituting for points $P_1 = [L_1\ 0]^T$, $P_2 = [0\ 0]^T$, $P_3 = [L_2\cos\theta_1\ L_2\sin\theta_1]^T$, and $P_4 = [L_2\cos\theta_1 - L_3\cos(\theta_1+\theta_2)\ L_2\sin\theta_1 - L_3\sin(\theta_1+\theta_2)]^T$, we get

$$P_s^1 = \frac{u_4 - u_3}{u_3 - u_1}\begin{bmatrix} -L_1 \\ 0 \end{bmatrix} \text{ and } P_s^2 = \frac{1}{u_5 - u_3}\begin{bmatrix} L_2\cos\theta_1(u_5-u_3) + L_3\cos(\theta_1+\theta_2)(u_3-u_2) \\ L_2\sin\theta_1(u_5-u_3) + L_3\sin(\theta_1+\theta_2)(u_3-u_2) \end{bmatrix}. \tag{25}$$

Since merging is done only when both the angles θ_1 and θ_2 cross the threshold angle, the limiting case can be taken as $\theta_1 = \theta_2 = \theta_m$. For equi-spaced knots, the above equation simplifies to

$$P_s^1 = \begin{bmatrix} -\frac{1}{2}L_1 \\ 0 \end{bmatrix} \text{ and } P_s^2 = \begin{bmatrix} L_2\cos\theta_m + \frac{1}{2}L_3\cos 2\theta_m \\ L_2\sin\theta_m + \frac{1}{2}L_3\sin 2\theta_m \end{bmatrix} \tag{26}$$

Fig. 9 shows the two points P_s^1 and P_s^2 that define the two control polygons respectively. We can make the following observations from the above analysis:

- If link lengths $L_1 = L_3 = L$, then in the limiting case mentioned above ($\theta_1 = \theta_2 = \theta_m$), the portion of spline shown in Fig. 9 is symmetric about an axis AA passing through the midpoint of points P_2 and P_3. The lengths of the two curves are equal and either of the two curves can be chosen after merging. The two curves and the length difference due to merging as a function of θ_m are shown in Fig. 10.

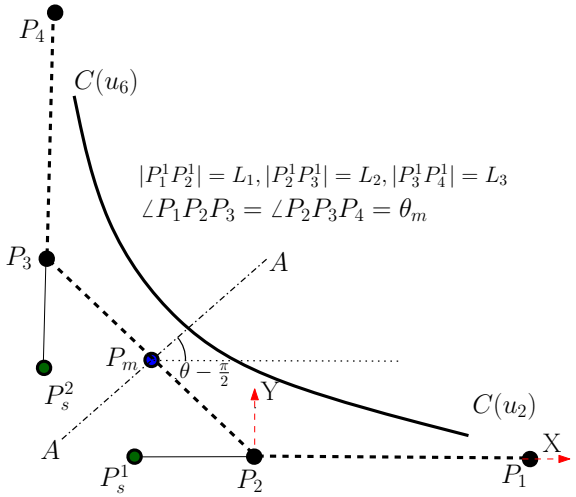
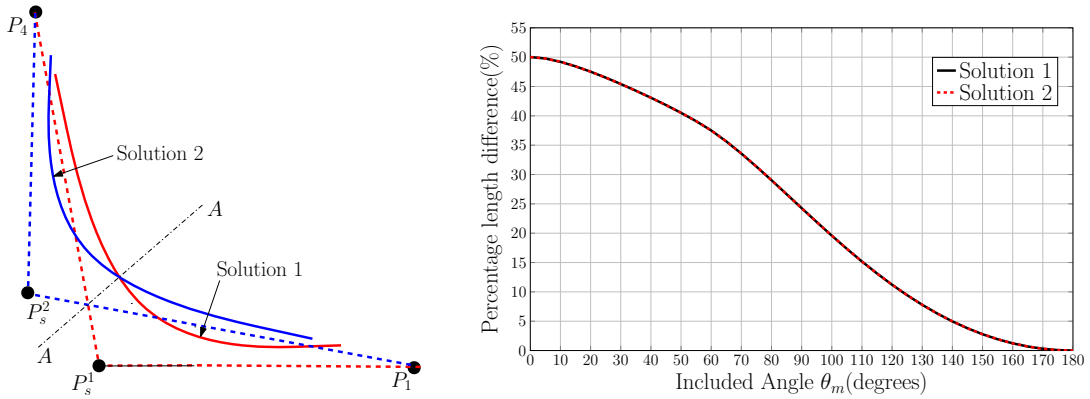


Figure 9: Portion of spline to be merged (knot removal)



(a) Affine nature of solutions

(b) Percentage length difference plot

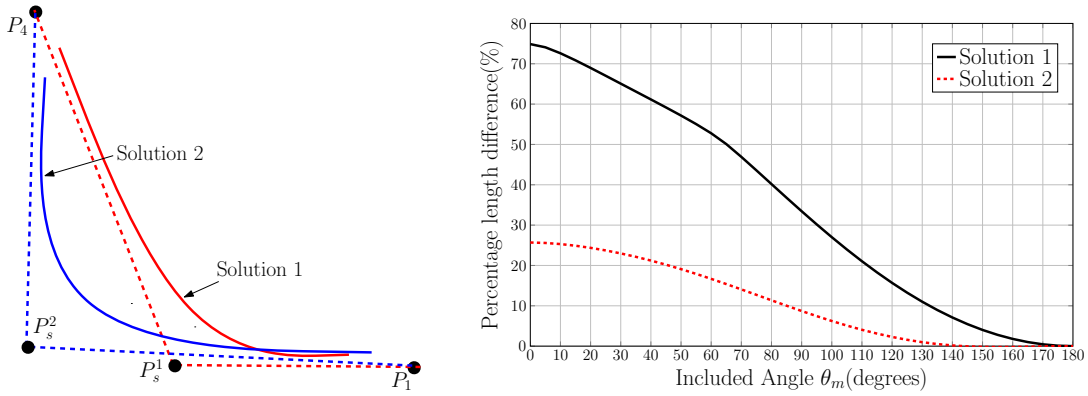
Figure 10: Multiple merging solutions when $L_1 = L_3$

- In a more generic case, where $L_1 \neq L_3$, the two solutions are not symmetrical. This is shown in Fig. 11 for arbitrarily chosen control polygon segment lengths $L_1 = 1$, $L_2 = 3$ and $L_3 = 4$. The difference between the curve length and the control polygon is shown in Fig. 11(b); during simulations we noted that the smaller of the two differences is chosen.

- The variation in length with change in included angle shown in figure 11 can be numerically computed and generated for other combination of lengths. Similar to subdivision, the plot of length difference versus θ_m can be used to choose the threshold angle for merging. From extensive numerical simulations (see also Section 4), it was observed that an angle of 160° for merging (knot removal) resulted in a total spline curve length error of less than 5%.

3.3 Algorithm for approximate length preservation

Based on the analysis in Sections 3.1 and 3.2, the algorithm followed for approximate length preserving configuration planning for a flexible 1D body is summarized in the flowchart of Fig. 12.



(a) Unsymmetric solutions

(b) Percentage difference plot

Figure 11: Multiple merging solutions when in a generic scenario

The inputs to the algorithm are the initial curve configuration, the path (curve) along which the leading end is moved, the velocity of the motion and location of the perturbed point on the curve. Based on these data, the initial control polygon configuration and the open uniform knot vectors are derived and initialized into the algorithm. The output is the motion of the flexible 1D object.

Steps of the Algorithm

1. Obtain the control polygon by interpolating the vertices/points in the input polyline. The number of segments in the control polygon is taken to be n, degree 3 and a clamped open uniform knot vector is used. The maximum deviation between the curve and the points is set to a predefined threshold for terminating the interpolation algorithm.

2. Initialize length parametrized motion of the leading end and discretized it for smooth visualization of motion.

3. For each incremental motion, starting from leading segment, recursively obtain the motion of all segments using the tractrix equations.

4. Perform threshold crossover checks for each included angle in the control polygon and execute subdivision or knot removal algorithms.

5. Update the new control polygon configurations and knot vectors and continue to next incremental motion of the leading end.

6. Repeat steps 3, 4 and 5 till the full motion of leading end is completed.

In the next Section, we present numerical simulation results illustrating the approach developed in this work.

4 Numerical simulation

We present numerical simulation results for a chosen 1D (1D) flexible object 5 units in length. All simulations were done using the commercial software MATLAB [29] on a PENTIUM quad core PC with 16 Gb RAM running the LINUX operating system. The first two simulations are for an arbitrarily chosen 2D path of the leading end and there are two 3D simulation. We present simulation

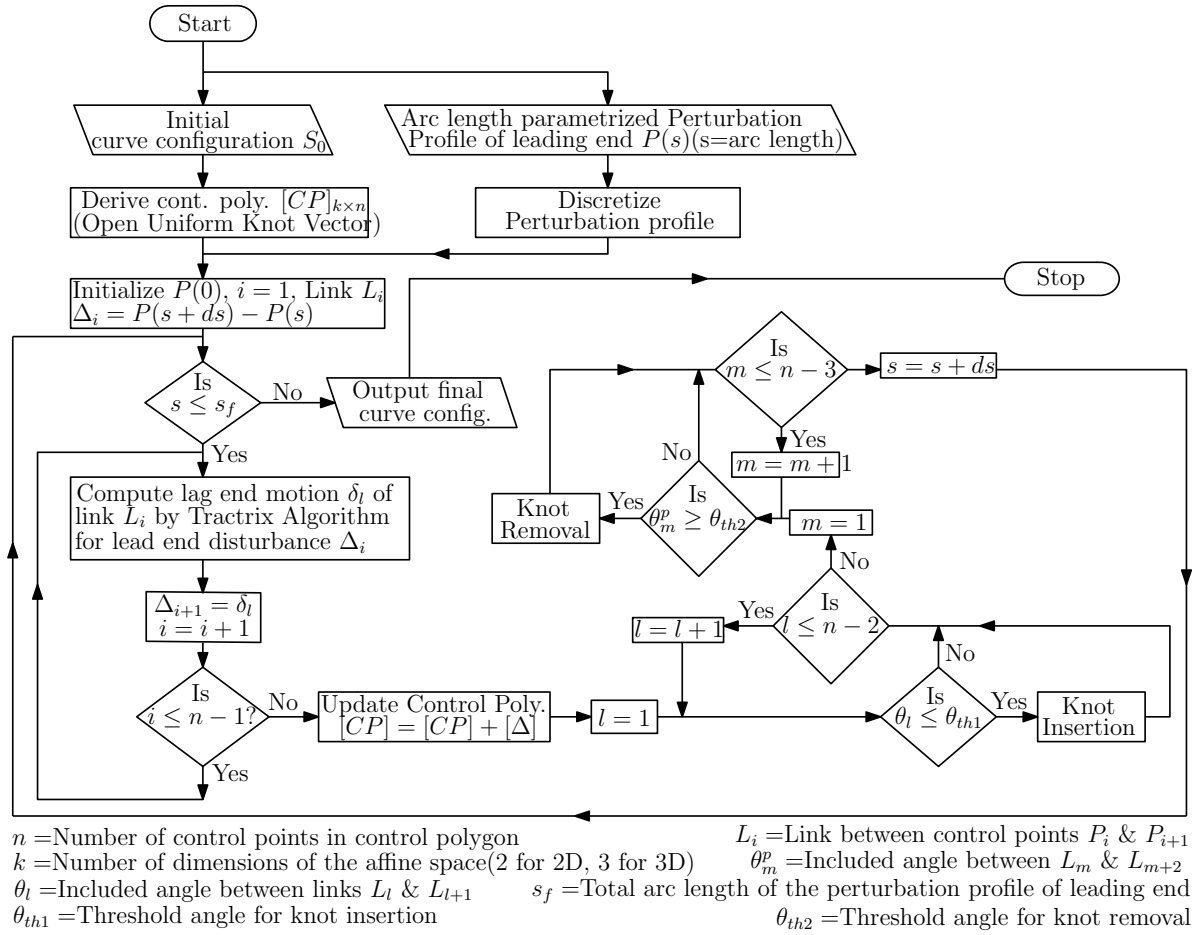
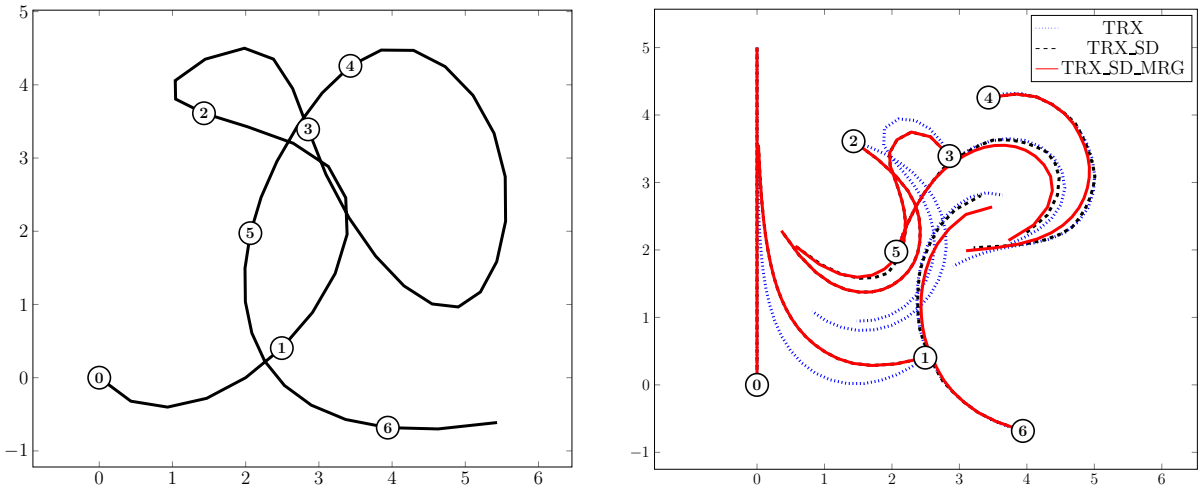


Figure 12: Flow chart of the algorithm

results for an exact length preserving tractrix based algorithm (labeled as TRX), an approximate length preserving tractrix and spline based algorithm with subdivision (labeled as TRX_SD) and finally an approximate length preserving tractrix and spline based algorithm with subdivision and knot removal (labeled as TRX_SD_MRG). In all the simulations the initial configuration of the flexible 1D object is a straight line – there is no restriction on the initial configuration and it is chosen as a straight line to ensure that the *initial* length error between the actual object and the discretized object/curve is zero.

In the first simulation, the leading end is moved along an arbitrarily chosen 2D path in steps of 0.5 length units over 52 steps. The arbitrary path is shown in Fig. 13 (a). Along the path, seven arbitrary snapshot locations are chosen and these are denoted by ⓪ through ⑥. The configuration of the flexible 1D object at each of the seven snapshot locations of the leading end is shown in Fig. 13 (b) – as mentioned at the start the flexible 1D object is a straight line (see configuration ⓪ in Fig. 13 (a)). The configuration of the flexible 1D object computed according to the three algorithms, namely TRX, TRX_SD and TRX_SD_MRG, are shown in Fig. 13 (b).

In the purely tractrix (TRX) algorithm, the flexible 1D object needs to be discretized into a large number of linear segments to realistically represent the continuous flexible 1D object and to make the motion look smooth. As the number of linear segments, n, increases, the computations, as expected, grow in a $\mathcal{O}(n)$ manner – the simulation time grows from 40 seconds for $n = 10$ to



(a) 2D path with snapshot locations

(b) Trajectory comparison with different algorithms

Figure 13: Simulation a planar curve subjected to generic 2D motion

about 1680 seconds for $n = 200$. In Fig. 13(b), the configuration simulations for the tractrix based simulations are done for $n = 20$.

For the approximate length preserving tractrix and spline based algorithms (TRX_SD and TRX_SD_MRG), the initial number of linear segments in the control polygon is chosen as 6 and the angle threshold for subdivision was chosen as 140°. The threshold angle for removing a control point/knot was chosen as 160°. Snapshots of the curve for different points on the chosen path are shown in Fig. 13(b) for illustrating the qualitative nature of the results. As mentioned the maximum percentage length difference depends on the chosen threshold angle (140° for subdivision and 160° for knot removal in this simulation) and the number of control points required to ensure that the angle threshold is not exceeded, in this case, turns out to be 20 for subdivision algorithm and 18 for subdivision with merging algorithm as seen from the Fig. 14. The maximum error introduced by merging at any instant over the whole simulation is 1.8%.

Fig. 14(b) shows the variation of curve length and Fig. 14(a) shows number of control points in the control polygon backbone over the simulation duration. As seen, the algorithm adapts to the characteristics of the motion by adding control points as and when required to compensate for the warping of the curve. It also removes control points as and when the curve can be simplified and represented in terms of a lesser number of control points. From the figures, it is observed that the maximum number of control points to represent the curve over the complete duration of simulation is 17 for TRX_SD_MRG algorithm.

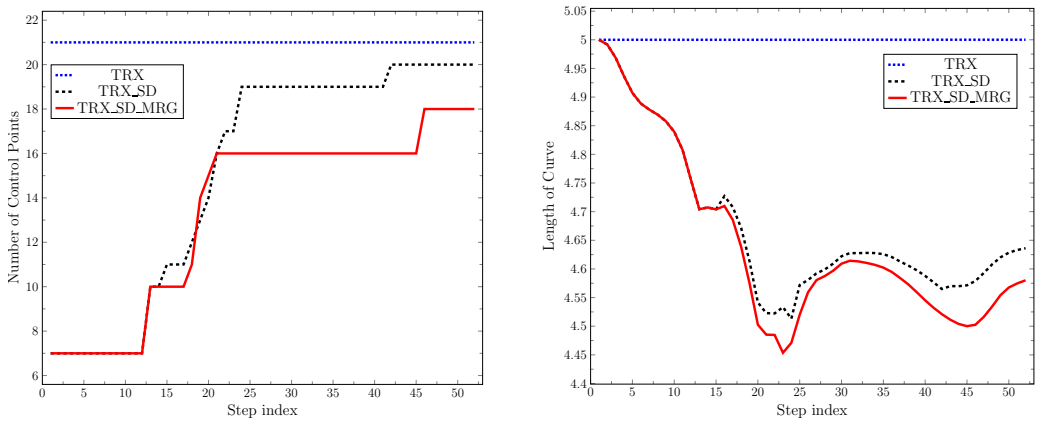
Intuitively, the higher the number of control points chosen initially, the lesser will be the error in length between control polygon and the spline. This is demonstrated in Table 1 in which results of TRX_SD_MRG simulation run on the planar curve input as 20 line segments is shown. As seen in the results, as the initial number of control points increases, the error in total length over the whole simulation comes down significantly to 4.90%. However, as seen in the results, the number of line segments in the backbone also increases accordingly. Finally, to see the effect of the threshold angle, we performed the TRX_SD_MRG simulations on the generic planar curve discretized into 20 segments with different threshold angles for subdivision and knot removal. The results are given in Table 2. Clearly, as the threshold angle increases, the curve sub-divides more frequently thereby resulting in more number of sides in the control polygon. However, the length error keeps on reducing with higher thresholds because the curve subdivides more to keep the length difference within bounds. The simulation results illustrate the theoretical results developed

No.	Initial number of points in Control Polygon	Max. % Error in Curve Length	Max. number of sides in Control Polygon	Total Simulation Time(sec.)
1	7	12.65	14	25.53
2	10	8.81	14	25.80
3	13	6.74	18	36.34
4	16	4.90	19	37.37

Table 1: Algorithm performance comparison for different initial number of control points in the control polygon

No.	Subdivision threshold(deg.)	Knot removal threshold(deg.)	Max. percentage diff. in length b/w curve and control polygon	Max. number of sides in control polygon	Total simulation time(sec.)
1	160	170	8.6	19	37.51
2	140	160	12.65	16	30.73
3	120	150	20.35	12	18.91

Table 2: Algorithm performance comparison for different threshold angles for subdivision and knot removal



(a) Variation in number of control points

(b) Variation in length of curve

Figure 14: Simulation a planar curve discretized by 50 linear segments subjected to generic 2D motion

earlier in Section 3. It may be mentioned that the actual numbers for length error and control points will vary based on the motion. However, the algorithm adaptively computes the lowest number of control points to effectively represent the curve respecting the prescribed error bounds. To illustrate this, a slightly modified trajectory of the leading end is chosen as shown in Fig. 15. In this trajectory at the end there exists a longer straight segment. The results for the simulation for the new motion are shown below and in the attached video (see file *example2D.mpg*). The number of segments initially chosen was 20 and the merging thresholds are relaxed to 150°. Other parameters were kept exactly the same as before. As expected, due to the presence of straight

trajectory towards the end, the number of control points further reduce to 12 when compared to 18 in the earlier case. However, due to relaxed merging threshold, the maximum error introduced by merging is now seen to be 4%. The numerical results shown in above figures and in the video

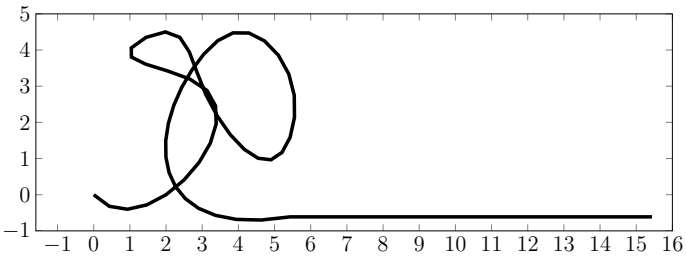
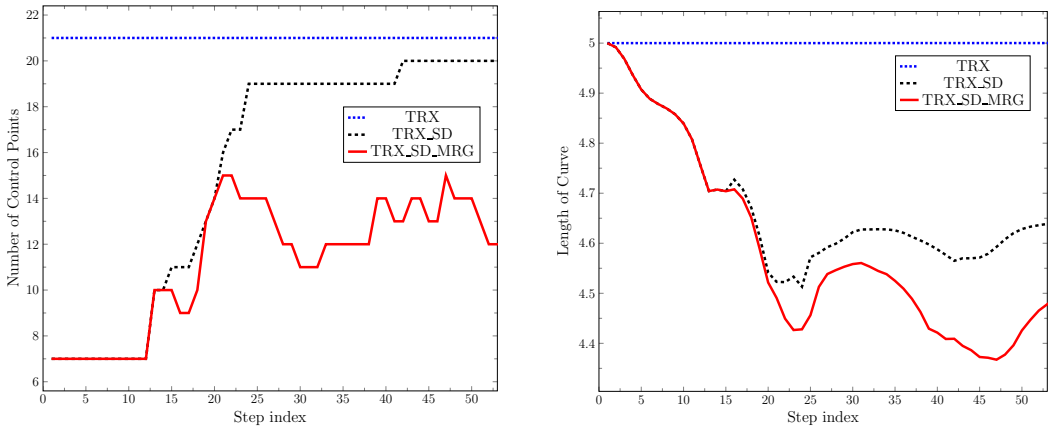


Figure 15: Modified motion trajectory for second simulation



(a) Variation in number of control points

(b) Variation in length of curve

Figure 16: Simulation of motion of a planar 1D curve with generic 2D motion at the leading end

clearly shows that the spline based algorithm results in a more natural motion together with a significant reduction in computation time. Additionally, the spline based algorithm is tunable for tolerances on length error specified by user requirements via the threshold values of the angles and the initial number of control points.

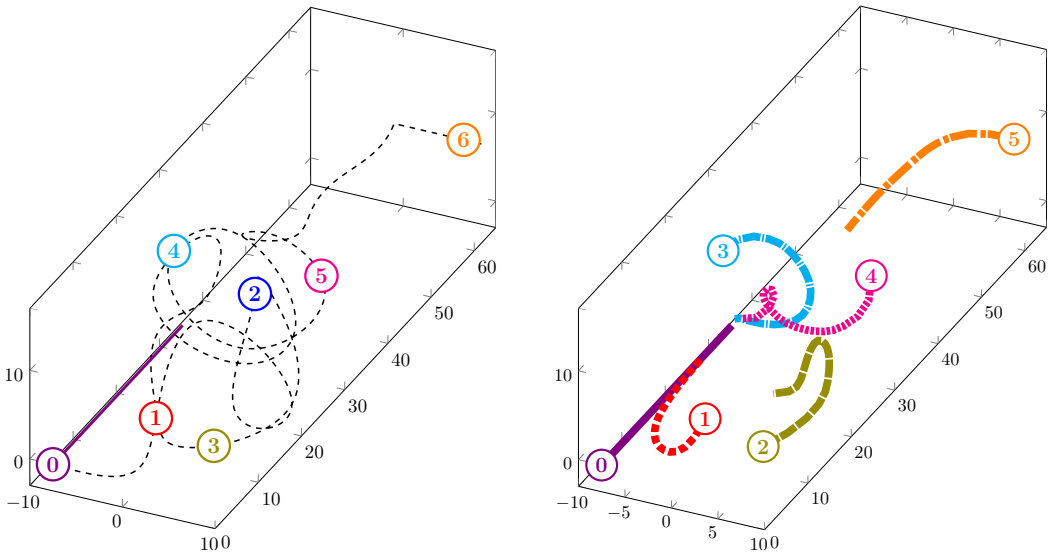
The algorithm is easily extendable to three dimensional space without any significant modifications. Fig. 17 shows the input desired motion of the leading end and simulated snapshots of motion of the complete flexible 1D object using the adaptive spline based subdivision algorithm proposed in this paper for a generic motion in three dimensions. In the attached video *excomp3D1.mpg*, three simulations are shown for a flexible 1D object of length 60 units and whose leading end is moved along an arbitrary path in 3D space. In the video, the left most simulation is for an exact length preserving tractrix based algorithm(TRX) where 15 linear segments are used to approximate the 1D flexible object. In the middle simulation, the number of linear segments is increased to 25 for enhanced smoothness and the right most simulation uses the approximate spline based algorithm, the path and snapshots of which are shown in Fig. 17. As shown in the video, the time for simulating with 15 segments is about 280 seconds while it increases to 914 seconds when 25 segments are used to make the motion appear more smoother. In comparison, the right most simulation shows an equally smooth motion with the approximate length preserving algorithm (difference less than 10%) in about 244 seconds. It may be noted that the approximate length preserving spline based

No.	Algorithm used	Initial number of points in control polygon	Maximum percentage difference between curve length and control polygon length	Total simulation time(sec.)
1	TRX	15	0.0	280
2	TRX	25	0.0	914
3	TRX_SD_MRG	12	8.9	244
4	TRX_SD_MRG	15	4.7	300

Table 3: Algorithm performance comparison in 3D

algorithm takes even smaller time than the exact length preserving algorithm with 15 segments as only a 12 sided control polygon is employed initially. As mentioned in the text, the length error can be brought down below 5% if more number of sides in the initial control polygon is used – if a 15 sided control polygon is used and time required is of the order of 300 seconds. Table 3 summarizes the simulation results for the 3D trajectory of Fig. 17.

A second video (see *excomp3D2.mpg*) shows similar improvement in execution time and smoothness in motion when the leading end is moved along a completely different arbitrary trajectory (Fig. 18(a)). Motion snapshots of this simulation are shown in Fig. 18(b).



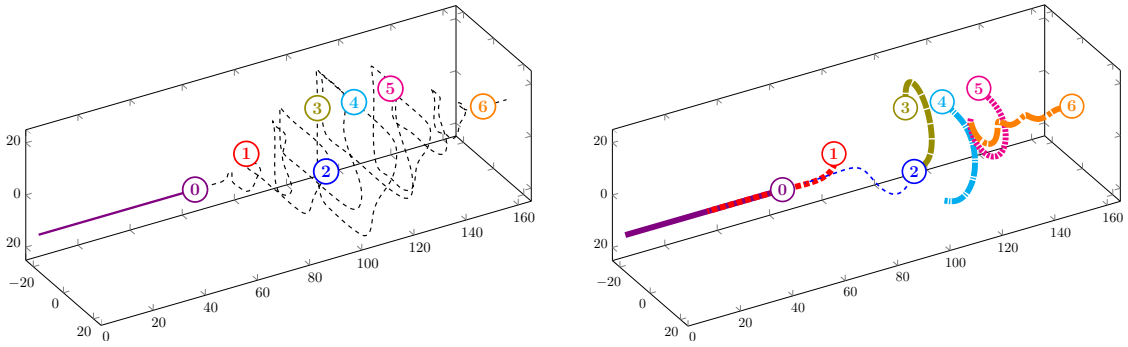
(a) A generic 3D motion for the leading end

(b) Motion snapshots along the path at various points

Figure 17: Motion simulation of a curve subjected to a generic 3D motion at the leading end (video file *excomp3D1.mpg*)

5 Conclusion

This paper proposes a new paradigm for the simulation and rendering of the motion of 1D flexible objects. The motion of the flexible 1D objects, represented using splines, is computed using a tractrix based approach. The tractrix based approach yields a more natural and realistic motion



(a) A generic 3D motion for the leading end

(b) Motion snapshots along the path at various points

Figure 18: Motion simulation of a curve subjected to a generic 3D motion at the leading end (video file *excomp3D2.mpg*)

with the motion dying out along the length of the flexible 1D object. The use of splines and adaptive modification of the control polygon leads to an approximate length preservation with efficient computation and smoother rendering of the motion of the 1D flexible object. An important feature of the proposed algorithm is that it is a purely *kinematics* or *geometry* based algorithm. It is shown that the use of splines and adaptive modification of the control polygon increases computation efficiency and the increased efficiency is more clearly observed when the number of segments in the input is large (more than 20 in the simulations shown in this work). The approach presented in this paper can be easily applied to simulation and realistic rendering of the motion of generic 1D flexible objects such as snakes, chains, ropes and for redundancy resolution in hyper-redundant robotic manipulators.

Appendix A

Quadratic spline with $L_1 \neq L_2$

For the general case of a quadratic spline with $L_1 \neq L_2$, the expression for the curve length is more complicated and is given by

$$
\begin{aligned}
l(\theta) &= \int_0^1 dl = \\
&\frac{(L_1^5 + 2L_1^3L_2^2 + 2L_1^2L_2^3 + L_2^5)\sqrt{L_1^2 + L_2^2 + 2L_1L_2\cos\theta} + (L_1^3L_2^2 + L_1^2L_2^3)\cos 2\theta\sqrt{L_1^2 + L_2^2 + 2L_1L_2\cos\theta}}{2(L_1^2 + L_2^2 + 2L_1L_2\cos\theta)^{\frac{5}{2}}} \\
&- \frac{L_1^2L_2^2(L_1^2 + L_2^2)\sin^2\theta}{2(L_1^2 + L_2^2 + 2L_1L_2\cos\theta)^{\frac{5}{2}}} \log \frac{-L_1^2 - L_1L_2\cos\theta + L_1\sqrt{L_1^2 + L_2^2 + 2L_1L_2\cos\theta}}{L_2^2 + L_1L_2\cos\theta + L_2\sqrt{L_1^2 + L_2^2 + 2L_1L_2\cos\theta}} \qquad \text{(A-1)} \\
&+ \frac{L_1L_2\cos\theta(3L_1^3 + L_1L_2^2 + L_2(L_1^2 + 3L_2^2))\sqrt{L_1^2 + L_2^2 + 2L_1L_2\cos\theta}}{2(L_1^2 + L_2^2 + 2L_1L_2\cos\theta)^{\frac{5}{2}}} \\
&+ \frac{2L_1^3L_2^3\sin^2\theta\cos\theta}{2(L_1^2 + L_2^2 + 2L_1L_2\cos\theta)^{\frac{5}{2}}} \log \frac{L_2^2 + L_1L_2\cos\theta + L_2\sqrt{L_1^2 + L_2^2 + 2L_1L_2\cos\theta}}{-L_1^2 - L_1L_2\cos\theta + L_1\sqrt{L_1^2 + L_2^2 + 2L_1L_2\cos\theta}}, \quad 0 < \theta < \pi.
\end{aligned}
$$

For case when $L_1 \neq L_2$ and $\theta \to 0$ (the curve folding and overlapping with itself), the curve length is given by $\lim_{\theta \to 0} l(\theta) = \dfrac{L_1^2 + L_2^2}{2(L_1 + L_2)}$. In the case of $L_1 \neq L_2$ and $\theta \to \pi$ (curve straightening out to a line), we have $\lim_{\theta \to \pi} l(\theta) = (1/2)(L_1 + L_2)$, which is another well known result for a quadratic B-spline curve (see pp. 78, [27]). From the analytical expression of $l(\theta)$ in Eq. (A-1), the maximum and minimum curve length occurs when $\theta \to \pi$ and when $\theta \to 0$, respectively.

Appendix B

Analyis for non-planar cubic spline

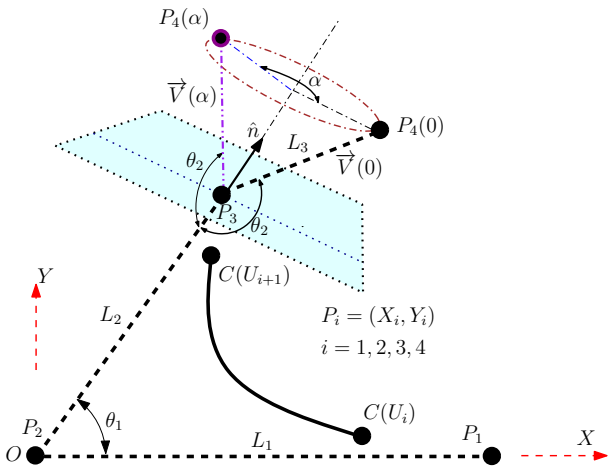


Figure 19: Three segments of the control polygon and planarity

Fig. 19 shows four points P_1, P_2, P_3, P_4 and the two included angles θ_1 and θ_2 between the three segments of a control polygon for a cubic spline. The first three points define a plane and the fourth point can lie anywhere on a cone with apex at P_3 and slant length $|P_3P_4|$ equal to L_3. We denote the vector on the surface of the cone along P_3P_4 by $\vec{V}(\alpha)$, where α is the angle of rotation about an unit vector $\hat{n}$ along the cone axis lying along P_2P_3. For $\alpha = 0$, the point P_4, denoted by $P_4(0)$, lies on the plane formed by P_1, P_2 and P_3 and the vector from P_3 to $P_4(0)$ is denoted by $\overrightarrow{V(0)}$. From Fig. 19, the angle between $\hat{n}$ and $\overrightarrow{V(0)}$ is seen to be $\pi - (\theta_1 + \theta_2)$ and we can write

$$\vec{V}(0) = [-L_3 \cos(\theta_1 + \theta_2) \;\; -L_3 \sin(\theta_1 + \theta_2) \;\; 0]^T . \tag{B-1}$$

The vector $\overrightarrow{V(\alpha)}$ can be obtained as

$$\overrightarrow{V(\alpha)} = R(\alpha)\overrightarrow{V(0)}, \tag{B-2}$$

where $R(\alpha)$ is a rotation matrix given by the Rodrigues' rotation formula

$$R(\alpha) = I_3 + (\sin\alpha)K + (1 - \cos\alpha)K^2, \tag{B-3}$$

with I_3 denoting a 3×3 identity matrix and $[K]$ given by

$$K = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix},$$

is a skew-symmetric matrix which represents the cross product operation with vector $\hat{n} = [n_1, n_2, n_3]^T$.

From Fig. 19, the vector $\hat{n}$ is given by $[\cos\theta_1, \sin\theta_1, 0]^T$ and we can write the vector locating the point $P_4(\alpha)$ with respect to the origin of the coordinate system O (or P_2) as

$$\overrightarrow{P_4}(\alpha) = \overrightarrow{P_3} + R(\alpha)\overrightarrow{V}(0) = \overrightarrow{P_4}(0) + \overrightarrow{\Delta P_4}(\alpha), \tag{B-4}$$

where

$$\overrightarrow{\Delta P_4}(\alpha) = \begin{bmatrix} L_3 \sin\theta_1 \sin\theta_2 (\cos\alpha - 1) \\ -L_3 \cos\theta_1 \sin\theta_2 (\cos\alpha - 1) \\ -L_3 \sin\theta_2 \sin\alpha \end{bmatrix}.$$

Using the above, the expression for the spline can be written as

$$C(u,\alpha) = N_{1,3}(u)\overrightarrow{P_1} + N_{2,3}(u)\overrightarrow{P_2} + N_{3,3}(u)\overrightarrow{P_3} + N_{4,3}(u)\left(\overrightarrow{P_4}(0) + \overrightarrow{\Delta P_4}(\alpha)\right), \tag{B-5}$$

and its derivative can be written as

$$\begin{aligned} C'(u,\alpha) = {} & N'_{1,3}(u)\begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix} + N'_{3,3}(u)\begin{bmatrix} L_2\cos\theta_1 \\ L_2\sin\theta_1 \\ 0 \end{bmatrix} \\ & + N'_{4,3}(u)\left(\begin{bmatrix} L_2\cos\theta_1 - L_3\cos(\theta_1+\theta_2) \\ L_2\sin\theta_1 - L_3\sin(\theta_1+\theta_2) \\ 0 \end{bmatrix} + \begin{bmatrix} L_3\sin\theta_1\sin\theta_2(\cos\alpha-1) \\ -L_3\cos\theta_1\sin\theta_2(\cos\alpha-1) \\ -L_3\sin\theta_2\sin\alpha \end{bmatrix}\right). \end{aligned} \tag{B-6}$$

When the spline lies in a plane and $\overrightarrow{\Delta P_4}(0) = 0$, the spline can be written as

$$C(u,0) = N_{1,3}(u)\overrightarrow{P_1} + N_{2,3}(u)\overrightarrow{P_2} + N_{3,3}(u)\overrightarrow{P_3} + N_{4,3}(u)\overrightarrow{P_4}(0), \tag{B-7}$$

and the derivative can be written as

$$C'(u,0) = N'_{1,3}(u)\begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix} + N'_{3,3}(u)\begin{bmatrix} L_2\cos\theta_1 \\ L_2\sin\theta_1 \\ 0 \end{bmatrix} + N'_{4,3}(u)\begin{bmatrix} L_2\cos\theta_1 - L_3\cos(\theta_1+\theta_2) \\ L_2\sin\theta_1 - L_3\sin(\theta_1+\theta_2) \\ 0 \end{bmatrix}. \tag{B-8}$$

The square of the elemental length of the cubic spline is given by

$$\begin{aligned} dl(\alpha)^2 = C'(u,\alpha)^T C'(u,\alpha)du^2 = {} & \\ & dl(0)^2 + 2N'_{4,3}(u)L_3\sin\theta_1\sin\theta_2(\cos\alpha-1)X'(u,0) + \left(N'_{4,3}(u)L_3\sin\theta_1\sin\theta_2(\cos\alpha-1)\right)^2 \\ & - 2N'_{4,3}(u)L_3\cos\theta_1\sin\theta_2(\cos\alpha-1)Y'(u,0) + \left(-N'_{4,3}(u)L_3\cos\theta_1\sin\theta_2(\cos\alpha-1)\right)^2 \\ & + \left(-N'_{4,3}(u)L_3\sin\theta_2\sin\alpha\right)^2, \end{aligned}$$

where $(X'(u,0), Y'(u,0))$ denote the X and Y components of $C'(u,0)$ – $Z'(u,0)$ is zero from Eq. (B-8). The above equation on simplifying gives

$$dl(\alpha)^2 = dl(0)^2 + 2N'_{1,3}N'_{4,3}L_1L_3 \sin\theta_1 \sin\theta_2(\cos\alpha - 1), \tag{B-9}$$

and by using the Cauchy-Schwarz inequality, we can write

$$dl(\alpha) - dl(0) \leq \sqrt{2N'_{1,3}N'_{4,3}L_1L_3 \sin\theta_1 \sin\theta_2(\cos\alpha - 1)}, \tag{B-10}$$

where $(\cos\alpha - 1) \leq 0 \ \forall\alpha \in [0\ 2\pi]$.

For a cubic spline, the basis functions $N_{1,3}$, $N_{4,3}$ and their derivatives are given by

$$N_{1,3} = \frac{1}{6}(1-u)^3,\ N'_{1,3} = -\frac{1}{2}(1-u)^2, N_{4,3} = \frac{1}{6}u^3,\ N'_{4,3} = \frac{1}{2}u^2, \tag{B-11}$$

and hence $dl(\alpha) - dl(0)$ is real and positive. Additionally, the right-hand side of Eq. (B-10) is maximum when $\alpha = \pi$ and minimum when $\alpha = 0$. The length of control polygon remains the same for all α and hence the elemental length difference between the cubic spline curve and the control polygon becomes maximum when $\alpha = 0$. This proves the assertion that the worst case difference between the length of the control polygon and the cubic spline is when $\alpha = 0$ (when the four control points lie on a plane) and we do not need to consider the four points in 3D space.

Acknowledgment

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

References

[1] Barzel, R. (1997), *Faking dynamics of ropes and springs*, *IEEE Computer Graphics Applications*, **17**(3), pp. 31-39.

[2] Hergenröether, E. and Däehne, P. (2002), *Real-time virtual cables based on kinematic simulations*, *Proceedings of WSCG 2000*, Czech Republic, February 7-11.

[3] Taskıran, H. D. and Güdükbay, U., (2005), *Physically-based simulation of hair strips in real-time. WSCG*, pages 153-156.

[4] Güdükbay, U., Özgüç, B. and Tokad, Y. (1997), *A spring force formulation for elastically deformable models. Computers & Graphics*, **21**(3), pp. 335-346.

[5] Natsupakpong, S. and Çavuşoğlu, M. C.(2010) *Determination of elasticity parameters in lumped element(mass-spring) models of deformable objects. Graphical Models*, **72**(6), pp. 61-73.

[6] Moll, M. and Kavaraki, L. E. (2006), *Path planning for deformable linear objects*, *IEEE Transactions on Robotics*, **22**(4), pp. 625-636.

[7] Ward, K., Bertails, F., Kim, T.-Y., Marschner, S. R., Cani, M.-P. and Lin, M. C (2007), *A survey on hair modeling: styling, simulation and rendering*, *IEEE Transactions on Visualization and Computer Graphics*, **13**(2), pp. 213-234.

[8] Grégoire, M. and Schömer, E. (2006), *Interactive simulation of one-dimensional flexible parts*, *Proceedings of 2006 ACM Symposium on Solid and Physical Modeling*, UK, June 6-8, pp. 95-103.

[9] Spillmann, J. and Tescner, M. (2009), *Cosserat nets*, *IEEE Transactions on Visualization and Computer Graphics*, **15**(2), pp. 325-338.

[10] Lenoir, J., Grisoni, L., Chaillou, C. and Meseure, P.(2005) *Adaptive resolution of 1D mechanical B-spline.*, In *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia GRAPHITE '05*, pp. 395-403, New York, USA..

[11] Theetten, A., Grisoni, L., Andriot, C. and Barsky, B.(2008) *Geometrically exact dynamic splines. Computer Aided Design*, **40**(1), pp. 35-48.

[12] Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M., and Grinspun. E (2007), *Efficient simulation of inextensible cloth*, *ACM Transactions on Graphics*, **26**(3), Art. no.49.

[13] Baraff, D. and Witkin, A. (1998), *Large steps in cloth simulation*, In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98*, New York, USA, pp. 43-54.

[14] Wang, H., O'Brien, J. F and Ramamoorthi, R (2010), *Multi-resolution isotropic strain limiting*, *ACM Transactions on Graphics(TOG)-Proceedings of ACM SIGGRAPH Asia 2010*, **29**(6), Art. no.156.

[15] Mikchevitch, A., Léon, J. C and Gouskov, A (2004), *Flexible beam part manipulation for assembly operation simulation in a virtual reality environment*, *Journal of Computing and Information Science in Engineering*, **4**(2), pp. 114-123.

[16] Nealen, A., Mller, M., Keiser, R. , Boxerman, E. and Carlson, M.(2006), *Physically based deformable models in computer graphics Computer Graphics Forum*, **25**(4), pp. 809-836.

[17] Brown, J., Latombe, J.-C. and Montogomery, K. (2004), *Real-time knot-tying simulation*, *The Visual Computer: International Journal of Computer Graphics*, **20**(2), pp. 165-179.

[18] Su, Z., Li, L. and Zhou, X. (2006), *Arc-length preserving curve deformation based on subdivision*, *Journal of Computational and Applied Mathematics*, **195**(1-2), pp. 172-181.

[19] Sreenivasan, S., Goel, P. and Ghosal, A. (2010), *A real-time algorithm for simulation of flexible objects and hyper-redundant manipulators*, *Mechanism and Machine Theory* **45**(3), pp. 454-466.

[20] Menon, M. S., Ananthasuresh, G. and Ghosal, A. (2013), *Natural motion of one-dimensional flexible objects using minimization approaches*, *Mechanism and Machine Theory*, **67**, pp. 64-76.

[21] Nakamura, Y. (1990), *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley Longman Publishing Co., Inc.

[22] Chirikjian, G. S. and Burdick, J. W.(1994), *A modal approach to hyper-redundant manipulator kinematics*, *IEEE Transactions on Robotics and Automation*, **10**(3), pp. 343-354.

[23] Reznik, D. and Lumelsky, V. (1994), *Sensor-based motion planning in three dimensions for a highly redundant snake robot*, *Advanced Robotics* **9**(3), pp. 255-280

[24] Ravi, V. C., Rakshit, S. and Ghosal, A.(2010), *Redundancy resolution using tractrix - Simulations and Experiments*, *Journal of Mechanisms and Robotics*, **2**(3), pp. 031013-031013-7

[25] Kühnapfel, U., Cakmak, H. K. and Maaß, H. (2000), *Endoscopic surgery training using virtual reality and deformable tissue simulation*, *Computers & Graphics*, **24**(5), pp. 671-682.

[26] Kang, H. and Wen, J. T. (2002), *Robotic knot tying in minimally invasive surgery*, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, **2**, pp. 1421-1426.

[27] Piegl, L. and Tiller, W. (1997), *The NURBS book*, Springer.

[28] Abbena, E., Salamon, S. and Gray, A. (2006), *Modern Differential Geometry of Curves and Surfaces with Mathematica*, CRC Press.

[29] MATLAB, version 7.12.0 (R2011a), *The MathWorks Inc.*, Natick, Massachusetts, 2012.