Studies on Motion and Deformation in Graphics

August 20, 2008

Studies on Motion and Deformation in Graphics

by

Dayue Zheng

A thesis submitted in partial fulfilment of the requirements for
the degree of Doctor of Philosophy
at the University of Hong Kong.

August 2008

Abstract of thesis entitled

Studies on Motion and Deformation in Graphics

Submitted by

Dayue Zheng

Abstract

In this thesis, we study the problems of deformation and motion in computer graphics. The studies begin a geometry problem on sweeping surface generation named rotation minimizing frame(RMF). We introduce a fourth-order approximate RMF algorithm named double reflection method. This solution can widely be used in strip object construction and realtime animation due to its precision, efficiency and robustness. We compare the new method with classical algorithms, and present further properties and extensions of the double reflection method for various application scenarios. Finally, we discuss the variational principles in design moving frames with boundary conditions, based on the RMF.

Then we discuss another problem in vector field deformation. A mesh surface in a vector field will receive force and perform deformation. Under a particular vector field construction, the deformation in the vector field will be automatically volume-preserving and non-intersecting. There are many interesting application due to these features. We extend the cross-product algorithm in [52] from a spherical field construction to a strip field construction. Moreover, we introduce a new curl vector field construction which is simpler and more efficient in vector field construction and realtime deformation. Compared with other vector field designs, this approach provides an easier and more intuitive construction for free-form objects.

Studies on Motion and Deformation in Graphics

by

Dayue Zheng

A thesis submitted in partial fulfilment of the requirements for
the degree of Doctor of Philosophy
at the University of Hong Kong.

August 2008

To My Family and Friends

Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signed:

Acknowledgements

I would like to express my gratitude to all people who help me to complete this thesis. First, I am much indebted to my supervisor Dr.Wenping Wang from The University of Hong Kong who helps and encourages me in all of my research during the four years. I also want to give my special thanks to Liu Yang and Bin Chan who are of great and kind help in my study and research. I Thank Jerry, Cathy as well as Rossi, life is a good time when being with you guys. Thanks group members Loretta, Dongming, Sunfeng, bob, Linda and Yufei, your kindness and support make it more than a research group. I also want to thank Linda for being an excellent partner of TA work in the Computer Graphics course. At last, I thank the Department of Computer Science for giving me the chance to enjoy the research life and complete my thesis in the University of Hong Kong.

Contents

1 Introduction I: RMF **1**

1.1 Background 1

1.2 Problem formulation 6

2 Summary of previous methods **8**

2.1 Frenet frame 8

2.1.1 general definition 8

2.1.2 Tangent vector, normal vector, binormal vector, curvature and torsion 9

2.1.3 Frenet-Serret formulas 11

2.2 Least rotation frame 11

2.2.1 Disadvantage of Frenet frame 11

2.2.2 parallel transport and least rotation frame 13

2.2.3 Computing the least rotation frame 13

2.3 Projection method 14

2.4 Rotation minimizing frame 16

2.5 The classical fourth-order Runge-Kutta method 17

3 Related Work **19**

3.1 Discrete approximation 19

3.2 Methods based on spine curve approximation 21

3.3 Numerical integration 22

3.4 The unstable torsion of a planar curve 24

4 Preliminaries **26**

4.1 Definition by differential equations 26

4.2 Some differential geometry 28

5 Double reflection method **31**

5.1 Outline of method 31

5.2 Geometric interpretation 35

5.3 Procedural description 38

5.4 Degenerate cases and symmetry 41

5.5 Invariance under conformal mappings 41

5.6 Order of approximation 42

5.7 **Proof of Theorem 5.6.1** 44

6 Comparison and Experiments **48**

6.1 Computational cost 48

6.2 Experimental results 50

7 Extensions **61**

7.1 Spine curve defined by a sequence of points 61

7.2 Using only tangent vectors 62

7.3 Variational principles for RMF with boundary conditions 63

8 Concluding remarks **73**

9 Introduction II: vector field deformation **74**

10 Related work II **77**

11 Cross-product vector field deformation **79**

11.1 Introduction of cross-product vector field construction 79

11.2 Extension of cross-product vector field construction 81

11.2.1 Extended spherical divergence free vector field 82

11.2.2 Divergence free vector field for strip: sticking action 83

11.2.3 Divergence free vector field for strip: sweeping action 84

11.2.4 Divergence free vector field for torus cases 86

11.2.5 Approximate vector field construction for free-form models . 87

11.2.6 Advanced discussion 89

12 Curl vector field deformation **92**

12.1 Translation 93

12.2 Rotation 96

12.3 Special application: twisting and bending 97

13 Implements and discussion **102**

13.1 Distance field computation 102

13.2 Remeshing 102

13.3 Experiments 104

13.4 Advantage and Limitations 104

13.5 Conclusion and Future Work 105

List of Tables

5.1 Algorithm — Double Reflection 40

6.1 The operations counts of the three sweeping methods. 50

6.2 Global approximation errors of the double reflection and 4-th Runge-kutta method for the torus knot. 54

6.3 Global approximation errors of the projection and rotation method for torus knot. 54

6.4 Global approximation errors of the double reflection and 4-th Runge-kutta method for the PH curve. 55

13.1 triangle numbers, volumes and frame rates in curl field deformation. 104

List of Figures

1.1 An example of using the RMF in shape modeling. 2

1.2 Sweep surfaces showing moving frames of a deforming curve. 5

2.1 Illumination of a Frenet frame 9

2.2 Illumination of least rotation frames 12

2.3 Illumination of frames in projection method 14

3.1 S-shaped sixth order Bézier curve 24

5.1 The first reflection $\mathcal{R}_1$ of the double reflection method. 33

5.2 The second reflection $\mathcal{R}_2$ of the double reflection method. 33

5.3 The projection of curve on sphere. 34

5.4 An RMF of a spherical curve. 34

6.1 Timings of the double reflection, projection and rotation method. . 51

6.2 Timings of Runge-Kutta method and the double reflection method. 51

6.3 Global errors of the four methods for the torus knot in Example 1. . 53

6.4 Global errors of the four methods for the PH curve in Example 2. . 53

6.5 Color surfaces showing the errors of double reflection method 56

6.6 Color surfaces showing the errors of 4-th order Runge-Kutta method 57

6.7 Color surfaces showing the errors of the projection method. 58

6.8 Color surfaces showing the errors of the rotation method. 59

6.9 The color coded sweep surfaces showing the errors for the PH curve. 60

7.1 Comparison in computing a closed moving frame. 67

7.2 comparison of different closing effect of a cinquefoil knot 69

7.3 comparison of different closing effect of a trefoil knot 70

7.4 Modeling of an oriental dragon. 71

7.5 A glass table: structure design based on sweep surfaces 72

9.1 Curl vector field deformation example 75

11.1 Translation and rotation field of cross-product method 80

11.2 Spherical vector field illustration. 82

11.3 Strip type-I vector field illustration. 83

11.4 strip type-II vector field illustration. 85

11.5 Torus vector field illustration. 86

11.6 Sphere, strip and free-form vector field deformation. 87

11.7 Comparison of sphere field, strip field and freeform field. 88

11.8 Deformation of Complicate objects: foot print on earth. 89

12.1 Illustration of vector flow in curl vector field. 92

12.2 Comparison of different divergence-free vector field construction. . . 94

12.3 Illustration of curl vector field generation for translational motion. . 95

12.4 Three collision examples: bunny, chess and rocker-arm. 95

12.5 Illustration of curl vector field generation for rotational motion. . . 97

12.6 vector flow of curl field and collision effect. 98

12.7 Simple rotational field generation. 98

12.8 Rectangular rod in curl vector field of twist effect. 99

12.9 Human body action: waist twist of Venus. 99

12.10Cube in curl vector field of bending effect. 100

12.11Tai Chi generated by curl vector field of bending effect. 100

12.12Deformation of complicate objects: hand print on earth. 101

Chapter 1

Introduction I: RMF

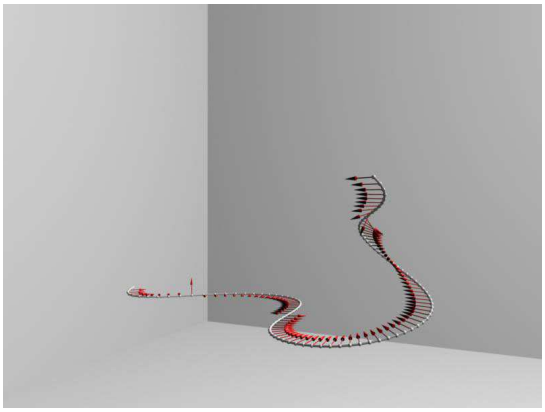
1.1 Background

Let $\mathbf{x}(u) = (x(u), y(u), z(u))^T$ be a C^1 regular curve in $\mathbb{E}^3$, the 3D Euclidean space. Denote $\mathbf{x}'(u) = \mathrm{d}\mathbf{x}(u)/\mathrm{d}u$ and $\mathbf{t}(u) = \mathbf{x}'(u)/||\mathbf{x}'(u)||$, which is the unit tangent vector of the curve $\mathbf{x}(u)$. We define a *moving frame* associated with $\mathbf{x}(u)$ to be a right-handed orthonormal system composed of an ordered triple of vectors $U(u) = (\mathbf{r}(u), \mathbf{s}(u), \mathbf{t}(u))$ satisfying $\mathbf{r}(u) \times \mathbf{s}(u) = \mathbf{t}(u)$. The curve $\mathbf{x}(u)$ in this context will be called a *spine curve*. Since $\mathbf{t}(u)$ is known and $\mathbf{s}(u) = \mathbf{t}(u) \times \mathbf{r}(u)$, a moving frame is uniquely determined by the unit normal vector $\mathbf{r}(u)$. Thus $\mathbf{r}$ is called the *reference vector* of a moving frame.

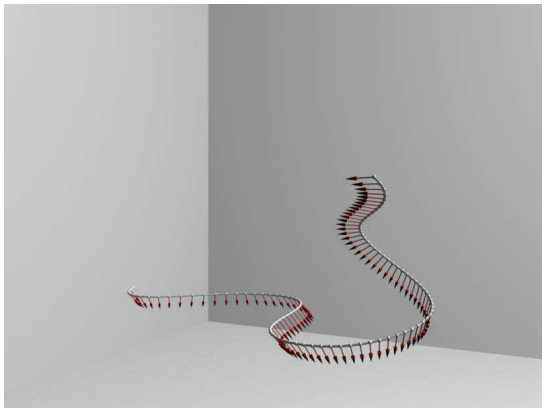
From the differential geometry point of view, a readily available moving frame of a curve in 3D is the Frenet frame, whose three orthogonal axis vectors are defined as

$$\mathbf{t} = \frac{\mathbf{x}'(u)}{\|\mathbf{x}'(u)\|}, \quad \mathbf{s} = \frac{\mathbf{x}'(u) \times \mathbf{x}''(u)}{\|\mathbf{x}'(u) \times \mathbf{x}''(u)\|}, \quad \mathbf{r} = \mathbf{s} \times \mathbf{t}.$$

Although the Frenet frame can easily be computed, its rotation about the tangent of a general spine curve often leads to undesirable twist in motion design or sweep surface modeling. Moreover, the Frenet frame is not continuously defined for a C^1 spine curve, and even for a C^2 spine curve the Frenet frame becomes undefined at



(a) The Frenet frame of a spine curve. Only the normal vector is shown.



(b) A rotation minimizing frame (RMF) of the same curve in (a). Only the reference vector is shown.

(c) A snake modeled using the RMF in (b).

Figure 1.1: An example of using the RMF in shape modeling.

an inflection point (i.e., curvature $\kappa = 0$), thus causing unacceptable discontinuity when used for sweep surface modeling [9].

A moving frame that does not rotate about the instantaneous tangent of the curve $\mathbf{x}(u)$ is called a *rotation minimizing frame* of $\mathbf{x}(u)$, or RMF, for short. It can be shown that the RMF is defined continuously for any C^1 regular spine curve. Because of its minimal-twist property and stable behavior in the presence of inflection points, the RMF is preferred to the Frenet frame in many applications in computer graphics, including free-form deformation with curve constraints [6, 39, 33, 35, 34], sweep surface modeling [10, 41, 46, 53], modeling of generalized cylinders and tree branches [45, 8, 13, 44], visualization of streamlines and tubes [3, 23, 21], simulation of ropes and strings [5], and motion design and control [28]. Discussion of the RMF and its applications can be found in the recent book by Hanson [22], where the RMF is treated using a parallel transport approach.

A typical application of RMF in shape modeling is shown in Figure 1.1. Here a canonical snake surface model is first defined along a straight line axis possessing an RMF generated by translation along the line. Then a new axis curve (i.e., a spine curve) is designed to produce a novel pose of the snake. For comparison, both Frenet frame and RMF of this same axis curve are shown in Figures 1.1(a) and 1.1(b). The RMF determines a mapping from the space of the canonical model of the snake to the space around the new axis curve in Figure 1.1(b); this mapping produces the snake in Figure 1.1(c). Note that the Frenet frame in this case exhibits excessive rotation compared with the RMF, so it is less appropriate for shape modeling.

Next consider moving frames of a deforming spine curve $\mathbf{x}(u;t)$, as frequently encountered in computer animation (see Figure 1.2). While the Frenet frame does not always experience abrupt twist for a given static spine curve, the Frenet frame

of the deforming spine curve often suddenly exhibits a radical twist at an instant during deformation, especially when the spine curve has a nearly curvature vanishing point (i.e., an inflection point). In contrast, the RMF of the deforming spine curve $\mathbf{x}(u;t)$ always varies smoothly and stably over time as well as along the spine curve. The different behaviors of these two moving frames are illustrated in Figure 1.2, visualized as sweep surfaces, through a sequence of snapshots of a deforming spine curve. Here by continuous deformation we mean that the rate of change in both position (i.e., $\partial\mathbf{x}(u;t)/\partial t$) and unit tangent (i.e., $\partial\mathbf{t}(u;t)/\partial t$) are bounded for any (u,t) in their finite intervals of definition. Note that, this assumption is reasonable in practical application but does not imply that the normal vector of $\mathbf{x}(u;t)$ changes continuously with respect to time t, thus explaining the potential instability of the Frenet frame.

Computation of the RMF is more involved than that of the Frenet frame. The RMF is first proposed and formulated as the solution of an ordinary differential equation in [7] and later in [45, 31]. Exact (i.e., closed form) RMF computation is either impossible or very involved for a general spine curve. Hence, a number of approximation methods have been proposed for RMF computation. These methods fall under three categories: 1) *discrete approximation*; 2) *spine curve approximation*; and 3) *numerical integration*. The discrete approximation approach is versatile for various applications in computer graphics and computer animation, even when only a sequence of points on a path (i.e., spine curve) is available, while the approach based on spine curve approximation is useful for surface modeling in CAGD applications. We will see that direct numerical integration of the defining ODE of RMF is relatively inefficient and therefore not well suited for RMF computation. The new method we are going to propose is based on discrete approximation.

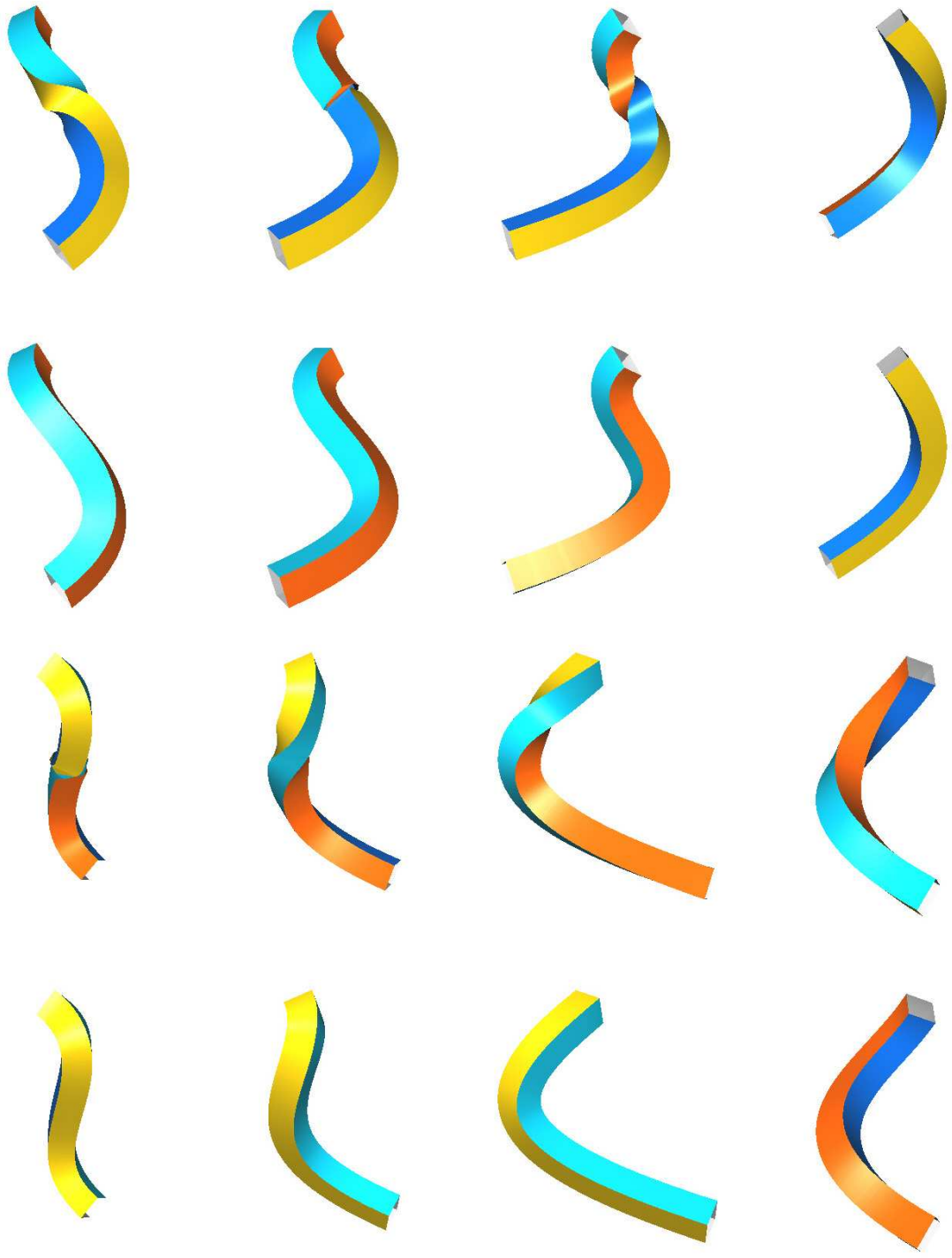


Figure 1.2: Sweep surfaces showing moving frames of a deforming curve:the Frenet frames in the first and the third rows and the RMF in the second and the forth rows.

1.2 Problem formulation

The RMF computation problem as solved by the discrete approximation approach is formulated as follows. Let $U(u)$ denote an exact RMF of a C^1 regular spine curve $\mathbf{x}(u)$ in 3D, $u \in [0, L]$, with the initial condition $U(0) = U_0$, which is some fixed orthonormal frame at the initial point $\mathbf{x}(0)$. Suppose that a sequence of points $\mathbf{x}_i = \mathbf{x}(u_i)$ and the unit tangent vectors $\mathbf{t}_i$ at $\mathbf{x}_i$ are sampled on the curve $\mathbf{x}(u)$, with $u_i = i * h$, $i = 0, 1, \ldots, n$, where $h = L/n$ is called the *step size*. The goal of discrete approximation is to compute a sequence of orthonormal frames U_i at $\mathbf{x}_i$ that approximates the exact RMF frame $U(u)$ at the sampled points, i.e., each U_i is an an approximation to $U(u_i)$, $i = 0, 1, 2 \ldots, n$.

Error measurement is needed to evaluate and compare different approximation schemes. Suppose that the exact RMF $U(u)$ has the same initial frame as the approximating frame sequence at $\mathbf{x}(u_0)$, i.e., $U(0) = U_0$. Then the approximation error between U_1 and $U(h)$ is called the *one-step error*. The approximation errors at intermediate sampled points are normally accumulated to give a large error at the end of the spine curve. However, due to error fluctuation, the maximum error may not always occur at the endpoint $\mathbf{x}(L)$. Therefore, we define the *global error* E_g to be the maximum error of frame approximation over all the sampled points $\mathbf{x}(u_i)$, i.e.,

$$E_g = \max_{i=0}^{n}\{\|U_i - U(u_i)\|\}, \tag{1.1}$$

where $\|U_i - U(u_i)\|$ is measured by the L_2 distance between the reference vectors $\mathbf{r}_i$ and $\mathbf{r}(u_i)$ of U_i and $U(u_i)$.

We shall present a new discrete approximation method, called *double reflection method*, for RMF computation. The main idea is based on the observation that the rigid transformation between two consecutive frames for RMF approximation

can be realized by two reflections, each being a reflection in a plane. The resulting method is simple, fast, and highly accurate – its global approximation error is of order $\mathcal{O}(h^4)$, where $h = L/n$ is the step size. This compares favorably with the second order (i.e., $\mathcal{O}(h^2)$) approximation error of two prevailing discrete approximation methods, i.e., the rotation method [9] and the projection method [31]. The accuracy of the double reflection method matches that of using the standard 4-th order Runge-Kutta method to integrate the defining differential equation of RMF, but is much simpler and faster than the latter.

In the following we will first review related works in Section 3 and present necessary preliminaries in 4. The double reflection method is presented and analyzed in Section 5. Then we present experimental verifications in Section 6, discuss extensions in Section 7 and conclude the paper in Section 8.

Readers interested only in implementation may skip to Section 5.1 for a simple description of the double reflection method; the pseudo code is given in Table 5.1 in Section 5.

Chapter 2

Summary of previous methods

Before we introduce our algorithm, for your briefing, we will discuss some relative sweeping algorithms first in this section.

2.1 Frenet frame

2.1.1 general definition

A Frenet frame is a reference frame system moving along given curves. It consists of n orthogonal vectors which usually represents a local coordinate system in n dimension space. Frenet frame contains many local properties such as curvature, torsion etc and it is more flexible in a local representation, so it is more widely used in differential geometric computing than global coordinate system like Euclidean coordinates.

In R^n space, the Frenet frame for a given C^{n+1} curve $c(t)$ is a set of of orthonormal vectors $e_1(t)$,$e_2(t)...e_n(t)$ We call them Frenet vectors and construct them from the derivatives of curve $c(t)$ using the Gram-Schmidt orthogonalization algorithm:

$$e_1(t) = \frac{c'(t)}{||c'(t)||}$$

$$e_j(t) = \frac{\overline{e_j}(t)}{||\overline{e_j}(t)||}$$

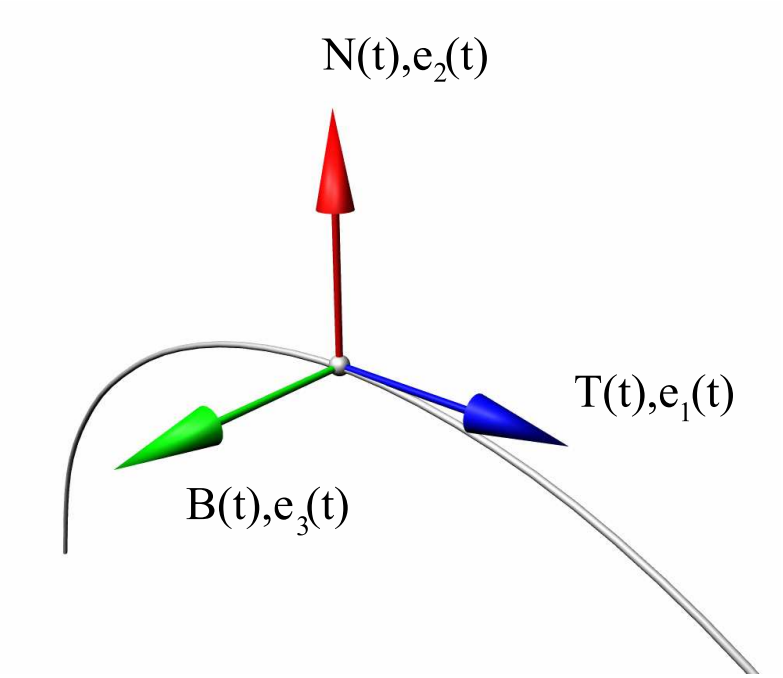


Figure 2.1: Frenet frame is a local coordinate reference system along curves. In 3-dimensions space, it contains three orthogonal unit vectors, $e_1(t)$,$e_2(t)$ and $e_3(t)$, which are also the tangent, normal and binormal on every point of curves.

$$\overline{e_j}(t) = c^{(j)}(t) - \sum < c^{(j)}(t), e_i(t) > e_i(t)$$

We call the real value function $x_i(t)$ **generalized curvatures**. It is a differential geometric property of the curve $c(t)$. Frenet frame and the generalized curvatures are invariant under reparametrization.

$$x_i(t) = \frac{<e'(t), e_{i+1}(t)>}{||c'(t)||}$$

2.1.2 Tangent vector, normal vector, binormal vector, curvature and torsion

In three-dimensional space, the first three Frenet vectors have particular names and construct current Frenet frame.

Mathematically, we define **tangent vector** as the instantaneous velocity at every point $P(t)|_{t=t_0}$ for a parameterized C^1 curve $c(t)$. Usually the curve represents the

path of a particle:

$$c'(t_0) = \frac{d}{dt}c(t)|_{t=t_0}$$

The first Frenet vector $e_1(t)$ is the **unit tangent vector** defined at each regular point of $c(t)$:

$$e_1(t) = \frac{c'(t)}{||c'(t)||}$$

If $t = s$ is the natural parameter then the tangent vector has unit length, so that the formula simplifies:

$$e_1(t) = c'(s)$$

The unit tangent vector determines the orientation of the curve, or the forward direction, corresponding to the increasing values of the parameter.

The **Normal vector** which is also called **curvature vector** indicates the deviance of the curve c(t):

$$\overline{e_2}(t) = c''(t) - < c''(t), e_1(t) > e_1(t)$$

After normalization, we get the second Frenet vector:

$$e_2(t) = \frac{\overline{e_2}(t)}{||\overline{e_2}(t)||}$$

Osculating plane are defined by the tangent and normal vectors.

The third Frenet vector $e_3(t)$ is **binormal vector** which is always orthogonal to the unit tangent and normal vectors:

$$\overline{e_3}(t) = c'''(t) - < c'''(t), e_1(t) > e_1(t) - < c'''(t), e_2(t) > e_2(t)$$

$$e_3(t) = \frac{\overline{e_3}(t)}{||\overline{e_3}(t)||}$$

Particularly, in 3-dimensional space the equation simplifies to

$$e_3(t) = e_2(t) \times e_1(t)$$

Curvature is the first generalized curvature $\chi_1(t)$ and measures the deviance of

c(t) from being a straight line:

$$\kappa(t) = \chi_1(t) = \frac{<e_1'(t), e_2(t)>}{||c'(t)||}$$

Torsion is the second generalized curvature $\chi_2(t)$ and measures the deviance of from being a plane curve:

$$\gamma(t) = \chi_2(t) = \frac{<e_2'(t), e_3(t)>}{||c'(t)||}$$

2.1.3 Frenet-Serret formulas

The Frenet-Serret formulas are constructed by ordinary differential equations of Frenet vectors and the generalized curvature functions χ_i

2-dimensions

$$\begin{bmatrix} e_1'(t) \\ e_2'(t) \end{bmatrix} = \begin{bmatrix} 0 & \kappa(t) \\ -\kappa(t) & 0 \end{bmatrix} \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix}$$

3-dimensions

$$\begin{bmatrix} e_1'(t) \\ e_2'(t) \\ e_3'(t) \end{bmatrix} = \begin{bmatrix} 0 & \kappa(t) & 0 \\ -\kappa(t) & 0 & \tau(t) \\ 0 & -\tau(t) & 0 \end{bmatrix} \begin{bmatrix} e_1(t) \\ e_2(t) \\ e_3(t) \end{bmatrix}$$

n-dimensions

$$\begin{bmatrix} e_1'(t) \\ ... \\ e_n'(t) \end{bmatrix} = \begin{bmatrix} 0 & \chi_1(t) & ... & 0 \\ -\chi_1(t) & 0 & ... & ... \\ 0 & ... & 0 & \chi_{n-1}(t) \\ 0 & ... & \chi_{n-1}(t) & 0 \end{bmatrix} \begin{bmatrix} e_1(t) \\ ... \\ e_n(t) \end{bmatrix}$$

2.2 Least rotation frame

2.2.1 Disadvantage of Frenet frame

Let's focus on the problem in 3-dimensions space. For curve c(s) parameterizing by arc length s from Parametric Equation c(u), we denote the Frenet frame as $F = (N(s), B(s), T(s))$

$$T(s) = c'(s), N(s) = \frac{T'(s)}{||T'(s)||}, B(s) = T(s) \times N(s)$$

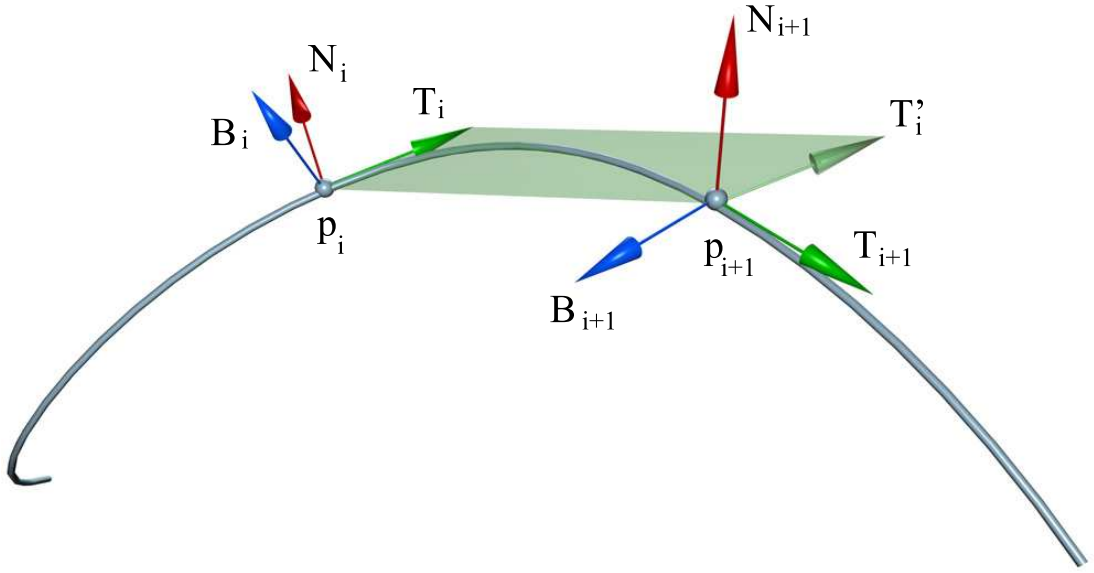


Figure 2.2: The main process of least rotation frame on each point: For a point p_{i+1}, The first frame vector T_{i+1} is the tangent of the curve. The second frame vector N_{i+1} is defined by the cross product of T_i and T_{i+1}. If T_i and T_{i+1} have the same direction, N_{i+1} will be defined as the same as N_i.

A computable Frenet frame required C^2 property. This definition is not available when a curve is less than C^2. Another important disadvantage is that Frenet frame contains intrinsically unnatural twist. We can measure it by torsion $\tau(s) = (dN/ds) \cdot B$:

$$N'(s) = (\frac{c''(s)}{||c''(s)||})' = -\frac{||c''(s)||c'''(s) - ||a''(s)||'a''(s)}{||a''(s)||^2}$$

In the bad case $||a''(s)||$ vanishes to give a point of inflection then the signs of both N and B change at the same time, the weird twist behavior happens. So Frenet frame is an unsatisfactory choice in geometric modeling though it is the most popular one.

2.2.2 parallel transport and least rotation frame

Least rotation frame is a quadratically convergent algorithm which can avoid the unnecessary twist created by the Frenet frame. The idea is that the tangent plane $P_{T(s)}$ should rotate by the minimum necessary to remain in it. This feature is supported by the process of **parallel transport** of tangent vectors of curves in R^n. Mathematically, **parallel transport** means, for a vector ν perpendicular to the tangent plane $P_{T(s)}$ on the conditions (I)$v \cdot T = 0$ and (II)$v' \perp P_{(s)}$, we can substitute it in $(\nu \cdot T)' = 0$ from $\rho = -T' \cdot \nu / T \cdot T \equiv -T' \cdot \nu / 1$, then we have

$$\nu(u) = -(T'(u) \cdot \nu(u))T(u)$$

Generally speaking, given $\overline{F} = (\overline{N}_1, \overline{B}_1, \overline{T}_1)$ and next tangent vector $\overline{T}_2$, the $\overline{N}_2$ and $\overline{B}_2$ in next frame can be defined as $\overline{N}_2 = \overline{T}_1 \times \overline{T}_2$ and $\overline{B}_2 = \overline{N}_2 \times \overline{T}_2$. The normal vector is constructed by the cross-product of two tangent vectors in order to keep rotation remain possible minimum. If the two tangent vectors point to the same direction that $T_1 = T_2$, Then next normal vector is defined as $N_2 = N_1$, which means the previous normal vector is used directly as current normal vector. No rotation happens in this condition.

2.2.3 Computing the least rotation frame

Numerical solution for the least rotation frame computing form $T(u)$ to $T(u+\Delta u)$ is given as follows. In this equation, $(a, b, c) = T(u) \times T(u+\Delta u)$ and $cos\alpha = T(u) \cdot T(u+\Delta u)$. Especially, if $\Delta u = (u_1 - u_0)/(n+1)$, we will have $R_n = \Pi_{i=0}^{n} R_{u_0 + i\Delta u, \Delta u}$ for a multiple rotation combination.

$$R_{u,\Delta u} = \begin{bmatrix} cos\alpha & -c & b \\ c & cos\alpha & -a \\ -b & a & cos\alpha \end{bmatrix} + \frac{1-cos\alpha}{a^2+b^2+c^2} \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 & bc \\ ac & bc & c^2 \end{bmatrix}$$

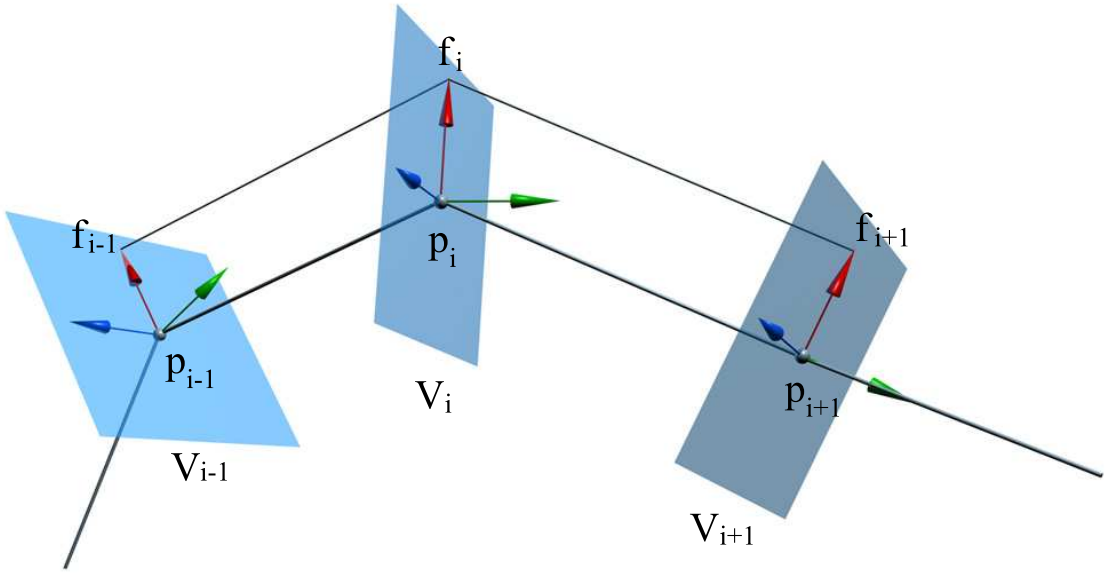


Figure 2.3: In projection method, current normal frame vector N_i is from the projection of previous normal frame vector N_{i-1}, which is projected to the plane passing current point p_i and the bisector of angle $\angle p_{i-1}p_ip_{i+1}$, orthogonal to the plane $p_{i-1}p_ip_{i+1}$.

2.3 Projection method

The projection method which is also named **rotation minimizing sweeps** for sweeping frame was presented by Klok in 1986. It also requires the trajectory curve $c(s)$ is a regular curve($||c'(s) \neq 0||$) as well as a non-restrictive assumption that c is arc-length parameterized($||c'(s) \equiv 1||$). Compared to Frenet frame, the projection method can choose the normal vector of frame at $c(0)$, when the normal vector of Frenet frame is determined by c''

Let's define the orthogonal frame $(t(s), f(s), g(s))$ along $c(s)$ with $t(0) = t_0$, $f(0) = f_0$, $g(0) = g_0$. $t(s) = c'(s)$ which denotes the tangent vector. Curve c is regular and $s \in [0, L]$. We also define a simple 2D contour $w(u), u \in [0, M]$, which is sweeping along curve $c(s)$. So the sweeping surface can be defined as

$$\Lambda(s,u) = c(s) + w_1(u)f(s) + w_2(u)g(s)$$

f(s) and g(s) is are the solution of the linear differential equation in ν: $\nu(s) = -(c''(s)\cdot\nu(s))c'(s)/||c'(s)||^2$ with initial conditions $f(0) = f_0$, $g(0) = g_0$. Moreover, the trajectory curve have a little more restriction for this algorithm. It must be regular and C^2 continuously differentiable with non-vanishing curvature.

Klok gave a construction of the approximation of this algorithm. This approximation has a main process of vector projection and it is the reason we name it **projection method**. First, we approximate the trajectory curve $c(s)$ by linear segments $p_0, p_1..., p_m$. $p_0, p_1..., p_m$ are the points on curve $c(s)$. We create a projection planes $V_0, V_1, ..., V_m$ on each point $p)i$ for projection process. f_i and g_i of orthogonal frame on p_i is on the projection plane V_i. Projection plane V_0 goes through p_0 and perpendicular to p_0p_1; projection plane V_m goes through p_m and perpendicular to $p_{m-1}p_m$; projection plane $V_i|_{i\neq 1,m}$ goes through p_i containing the bisector of angle $\angle p_{i-1}p_ip_{i+1}$ and orthogonal to the plane $p_{i-1}p_ip_{i+1}$. For the first frame vector $t(s)$, t_0 is the same direction as p_0p_1, when t_m is the same direction as $p_{m-1}p_m$. $t_i|_{i\neq 1,m}$ locates on the plane $p_{i-1}p_ip_{i+1}$ and perpendicular to the bisector of $\angle p_{i-1}p_ip_{i+1}$. If we use curve c(s) directly, $t(s) = c'(s)$. For the second frame vector $f(s)$, we choose f_0 freely. When we define f_i in vector sequence $f_0, f_1, ..., f_m$, we project it to next projection plane V_{i+1} and get the project vector $\overline{f_i}$. Then f_{i+1} is defined as a unit vector and has the same direction as $\overline{f_i}$. This is the projection process. After all, we define $g_i = f_i \times t_i$. So the orthogonal frame for each point p_i is constructed. Based on these orthogonal frames, translational sweepings is able to be created.

2.4 Rotation minimizing frame

For twice differentiable curve c(s), we call the orthogonal frame $(t(s), f(s), g(s))$ to be **rotation minimizing frame**: $t(s)$ denotes the tangent vector of curve c(t); f(s) and g(s) is are the solution of the linear differential equation in ν, which has initial conditions $t(0) = t_0$, $f(0) = f_0$, $g(0) = g_0$:

$$\nu(s) = -(c''(s) \cdot \nu(s))c'(s)/||c'(s)||^2$$

Rotation minimizing frame is a good solution for sweeping problems. In the following rotation minimizing frame will be referred to RMF. Actually, projection method is C^2 approximation of RMF method.

RMF method can also be approximately computed by integral formula. For a trajectory curve $c(u)$, we define the RMF as $(X(u), Y(u), Z(u))$, in which $Z(u) = c'(u)/||c'(u)||$, $X(u)$ and $Y'(u)$ can be defined as

$$X(u) = cos\theta(u)N(u) + sin\theta(u)B(u)$$

$$Y(u) = -sin\theta(u)N(u) + cos\theta(u)B(u)$$

with

$$\theta(u) - \theta_0 = -\int_{u_0}^{u} \tau(t)||c'(t)||dt$$

$\tau(u)$ is the torsion of curve c(u):

$$\tau(u) = \frac{det(c'(u), c''(u), c'''(u))}{||c'(u) \times c''(u)||^2}$$

Torsion represents the twist of curves. So the integrated torsion of curve $c(u)$ is the same as the amount of twist of the Frenet frame. In this method, we create RMF by apply the additional twist rotation to each Frenet frame. Since integral formula is not easy to computed directly, we use numerical integration to compute approximate $\theta(u)$. In the equation of torsion $\tau(u)$, $c'(u) \times c''(u$ is unstable when $c'(u) = 0$ or $c''(u) = 0$. RMF is rotated from Frenet frame in this approximated

method, so it has the same unrobust twist problem as Frenet frame.

2.5 The classical fourth-order Runge-Kutta method

We use the classical fourth-order Runge-Kutta method to do the numerical computing for rotation minimizing frame method in the thesis. Runge-Kutta methods are developed by German mathematicians C.Runge and M.W.Kutta around 1900. They are an important set of implicit and explicit iterative methods for the approximation of solutions of ordinary differential equations. Runge-Kutta method family contains classical Runge-Kutta methods, explicit Runge-Kutta methods, adaptive Runge-Kutta methods, implicit Runge-Kutta methods, etc. classcical Runge-Kutta methods, especially the classical fourth-order Runge-Kutta method is the most popular one and commonly used in numerical analysis.

First lets specify an initial value problem as follows:

$y' = f(t, y),\ y(t_0) = y_0$

Then the classical fourth-order Runge-Kutta method for the problem is given by the equations:

$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$t_{n+1} = t_n + h$

The four parameters are specified as

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ k_4 = f(t_n + h, y_n + hk_3) \end{cases}$$

k_1 is the slope at the beginning of the interval and k_4 is the slope at the end. k_2 and k_3 are both the slope at the midpoint $t_n + \frac{h}{2}$ using Euler's method but determined by different k_i.

The error per step of classical fourth-order Runge-Kutta method has order h^5, while the total accumulated error have order h^4.

Chapter 3

Related Work

3.1 Discrete approximation

In discrete approximation an RMF is approximated by a sequence of orthogonal frames located at sampled points $\mathbf{x}_i$ on the spine curve $\mathbf{x}(u)$. The projection method, as originally proposed in [31], computes an approximate RMF for modeling a sweep surface. Suppose that the the sampled points $\mathbf{x}_i$ and the unit tangent vectors $\mathbf{t}_i$ of $\mathbf{x}(u)$ at the sampled points $\mathbf{x}_i$ are provided as input. For RMF computation, the projection method projects, along the direction $\mathbf{x}_1 - \mathbf{x}_0$, an initial reference vector $\mathbf{r}_0$ in the normal plane of the spine curve at $\mathbf{x}_0$ to the next reference vector $\mathbf{r}_1$ on the normal plane at $\mathbf{x}_1$. Then this step is repeated to generate on the subsequent normal planes a sequence of reference vectors $\mathbf{r}_i$, which, together with the tangent vectors $\mathbf{t}_i$, define a sequence of orthonormal frames that approximate an exact RMF. The projection method is empirically demonstrated to have the second order of approximation error [15]. Note that the above projection between normal planes is not length preserving. Therefore the reference vectors $\mathbf{r}_i$ need to be normalized to give unit vectors.

Another popular discrete approximation method is the *rotation method* [9, 46, 40]. The rotation method also needs as input the sampled points $\mathbf{x}_i$ on the spine

curve and the unit tangent vectors $\mathbf{t}_i$ of the spine curve at $\mathbf{x}_i$. Consider the first two sampled points $\mathbf{x}_0$ and $\mathbf{x}_1$. Given the initial frame U_0 at $\mathbf{x}_0$, suppose that we need to compute the next frame U_1 at $\mathbf{x}_1$ from the *boundary data* $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$. To minimize the rotation about the tangent of the spine curve, this method rotates U_0 into U_1 about an axis $\mathbf{b}_0$ perpendicular to $\mathbf{t}_0$ and $\mathbf{t}_1$, that is, $\mathbf{b}_0 = \mathbf{t}_0 \times \mathbf{t}_1$; the rotation angle θ is such that the frame vector $\mathbf{t}_0$ of U_0 is brought into alignment with the frame vector $\mathbf{t}_1$ of U_1, i.e., $\theta = \arccos(\mathbf{t}_0 \cdot \mathbf{t}_1)$. Here, for frame computation, we ignore the translational difference between the origins of U_0 and U_1. The rotation method has the second order global approximation error [40].

A major problem with the rotation method is its lack of robustness for nearly collinear data. When the two consecutive tangent vectors $\mathbf{t}_0$ and $\mathbf{t}_1$ are collinear, the rotation axis becomes undefined, since $\mathbf{b}_0 = \mathbf{t}_0 \times \mathbf{t}_1 = 0$; but, since no rotation is needed in this case, we just need to set $U_1 := U_0$. However, numerical problems will be experienced when $\mathbf{t}_0$ and $\mathbf{t}_1$ approach each other, i.e., becoming closer and closer to being collinear; this happens, for example, when the spine curve is densely sampled for high accuracy RMF computation. In this case some threshold value has to be used to avoid the degeneracy of the rotation vector $\mathbf{b}_0$ by treating nearly collinear data as collinear data. But if a spine curve is so densely sampled that all consecutive data segments are deemed as collinear due to thresholding, then there will be a large accumulated error in the computed RMF, because the spine curve will be treated as a straight line and all the frames U_i will be set to be identical to the initial frame U_0. We note that this numerical problem for nearly collinear data does not exist with the double reflection method we are going to propose.

3.2 Methods based on spine curve approximation

If a spine curve is first approximated by some simple curves whose RMF can be computed exactly or more accurately, then the RMF of this simple approximating curve can be taken as an approximation to the RMF of the original spine curve. An intuitive argument for this idea is that if two spine curves are close to each other, then their RMFs should also be. This type of intuition lacks rigorous justification and could be unreliable for moving frames defined by differential properties; recall that the Frenet frames of two spine curves close to each other can be radically different. A related result by Poston et al [40] basically states that the RMF of a spine curve $\tilde{\mathbf{x}}(u)$ approaches the RMF of another spine $\mathbf{x}(u)$ if and only if the unit tangent vector $\tilde{\mathbf{t}}(u)$ of $\tilde{\mathbf{x}}(u)$ approaches the unit tangent vector $\mathbf{t}(u)$ of $\mathbf{x}(u)$.

Discrete approximation methods, such as the projection method or the rotation method, can be regarded as the simplest methods based on spine curve approximation, using a polygon to approximate the spine curve. A G^1 spline curve composed of circular arcs is used to approximate an input spine curve in [53] to compute an approximate RMF for modeling sweep surfaces in NURBS form. The spine curve is approximated by PH curves using Hermite interpolation in [29] for generating sweep surfaces in rational representation. Exact description of the RMF of a PH curve and its rational approximation are provided in [28, 18, 19, 14]. A closely related technique is to approximate the rotation minimizing motions (RMM) by affine motions (cf. [41]) and rational motions from the point of view of spherical kinematics [27].

3.3 Numerical integration

Since the RMF is defined by a vector-valued ODE of the type $\mathbf{y}' = \mathbf{f}(\mathbf{x}, \mathbf{y})$ [7, 45, 31, 41], naturally one may consider computing the RMF using a numerical method to directly solve this ODE. Suppose that the classical fourth order Runge-Kutta method is used. Then the RMF thus computed has the 4-th order global approximation error, which is the same as that of the double reflection method that we are to propose. However, this general approach to solving the ODE does not take into account the special geometric property of the problem of RMF computation and therefore has severe drawbacks.

Firstly, the Runge-Kutta method requires the spine curve $\mathbf{x}(u)$ to be C^2, since the right hand side $\mathbf{f}$ of the ODE is a function of the second derivative of $\mathbf{x}(u)$ (cf. Eqn. (4.3) in Section 4). This requirement is unnecessarily restrictive, since the RMF is continuously defined for any C^1 spine curve. Secondly, deriving and evaluating the second derivative of $\mathbf{x}(u)$ can be tedious and costly, rendering the method inefficient. In the RMF computation problem under consideration, the sampled points $\mathbf{x}_i$ and the tangent vectors $\mathbf{t}_i$ are available as input. But both first and second derivatives of the spine $\mathbf{x}(u)$ are required by the Runge-Kutta method. This mismatch between the input data of the RMF computation problem and the data it requires makes the Runge-Kutta method not well suited for RMF computation.

Another problem is that the Runge-Kutta method does not strictly enforce the orthogonality between the solved reference vectors $\mathbf{r}_i$ and the tangent vectors $\mathbf{t}_i$, even though in the initial conditions $\mathbf{r}_0 = \mathbf{r}(0)$ is orthogonal to $\mathbf{t}_0 = \mathbf{t}(0)$. Therefore each $\mathbf{r}_i$ has to be projected onto the normal plane of the spine curve to make it perpendicular to $\mathbf{t}_i$; this adds further to the cost of the method.

Another method is based on the observation that the RMF and the Frenet frame differ by a rotation determined by the torsion in the normal plane of the spine curve. Let $\theta(u)$ be the angle of this rotation. Let $\tau(u)$ be the torsion of the spine curve $\mathbf{x}(u)$. Then $\theta(u)$ is given by [20]

$$\theta(u) = -\int_{u_0}^{u} \tau(v)\|x'(v)\|\mathrm{d}v \tag{3.1}$$

With this formula, $\theta(u)$ may be computed with some quadrature rule and used to compute the RMF by compensating the rotation of the Frenet frame. However, at inflection points of a spine curve, the Frenet frame itself becomes discontinuous and exhibits abrupt change, and the torsion $\tau(u)$ becomes ill-defined (i.e., unbounded), making it difficult to evaluate the integration (3.1) accurately; therefore in this case the method becomes unstable. This problem is further discussed with an example in next section.

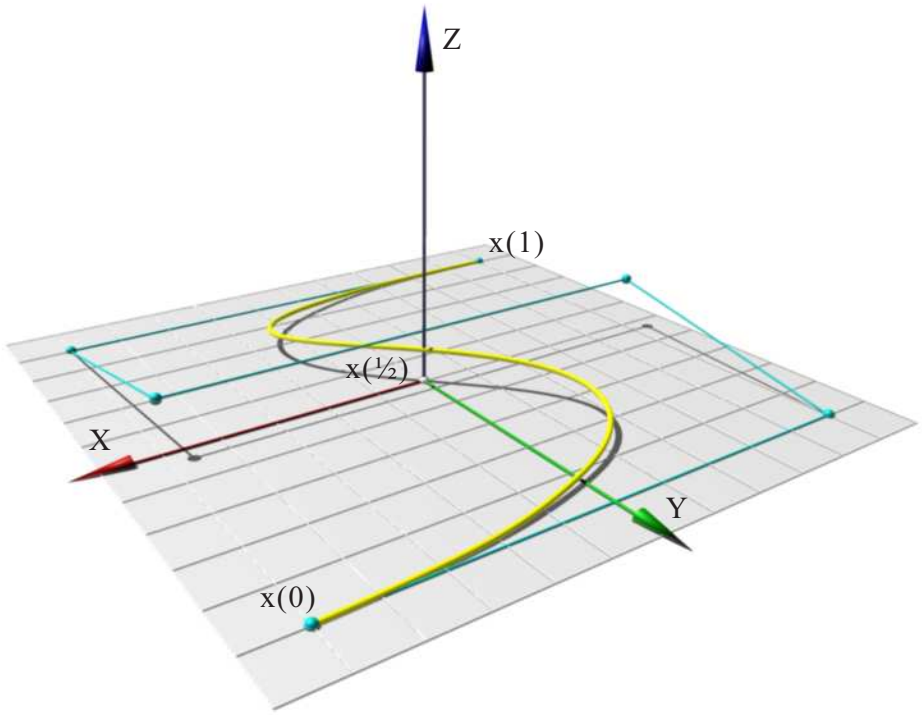


Figure 3.1: S-shaped sixth order Bézier curve

3.4 The unstable torsion of a planar curve

It is well known that a curve is planar if and only if its torsion is zero everywhere. However, the torsion has certain peculiar behavior which makes it an "unstable" characteristic of a planar curve. Below we use an example to show that a nearly planar curve can have arbitrarily large torsion. This example also serves two further purposes. First, the numerical integration method in [20] will experience severe difficulty in computing the RMF for the spine curve in this example. Second, although the torsion τ is unbounded in this example, we will see that the constant K in the fifth order error term in Theorem 5.6.1 is still finite, thus showing that the second part of the proof of Theorem 5.6.1 is warranted (cf. Appendix I).

Consider the S-shaped sixth order Bézier curve $\mathbf{x}(t;h)$ in 3D defined by the

control points $P_0 = (1,1,0)^T$, $P_1 = (-1,1,0)^T$, $P_2 = (-1,0,h)^T$, $P_3 = (1,0,h)^T$, $P_4 = (1,-1,0)^T$ and $P_5 = (-1,-1,0)^T$. The curve has the parametric equation

$$\mathbf{x}(t;h) = \begin{pmatrix} 8t^5 - 20t^4 + 20t^2 - 10t + 1 \\ 8t^5 - 20t^4 + 20t^3 - 10t^2 + 1 \\ 10t^4h - 20t^3h + 10t^2h \end{pmatrix}, \quad t \in [0,1]$$

We consider the behavior of $\mathbf{x}(t;h)$ as $h \to 0$. When $h = 0$, $\mathbf{x}(t;0)$ becomes planar and $\mathbf{x}(1/2;0) = (0,0,0)$ is an inflection point.

Let us first check $\tau_0, \tau_1, \kappa_0, \kappa_1, \kappa_2$ at the point $\mathbf{x}(1/2;h)$.

$$\tau_0 = -\frac{12}{5h}, \quad \tau_1 = 0, \quad \kappa_0 = \frac{4}{5}|h|, \quad \kappa_1 = 0, \quad \kappa_2 = \frac{192(h^4 - 3h^2 - 3)}{125|h|}$$

Clearly, τ_0 and κ_2 are not bounded as $h \to 0$.

Now we check K at $\mathbf{x}(1/2;h)$. The four terms of the expression of K in Eqn. (5.7) are

$$2\kappa_1^2\tau_0 = 0, \quad \kappa_0^2\tau_0^3 = -\frac{27648}{3125h}, \quad \kappa_1\kappa_0\tau_1 = 0, \quad \kappa_2\kappa_0\tau_0 = \frac{9216(h^4 - 3h^2 - 3)}{3125h}$$

Then

$$K = 2\,{\kappa_1}^2\tau_0 + {\kappa_0}^2{\tau_0}^3 + \kappa_1\kappa_0\tau_1 - \kappa_2\kappa_0\tau_0 = \frac{9216h(3 - h^2)}{3125}$$

and $\lim_{h\to 0} K = 0$. Hence, K is finite for all finite values of h in this example.

In this example, numerical integration of (3.1) becomes difficult as the integrand $\tau(t)$ becomes unbounded at $t = 1/2$ for arbitrarily small h. Furthermore, even if this integration can be done, the Frenet frame of $\mathbf{x}(t)$ becomes increasingly unstable at $t = 1/2$ as $h \to 0$. All this makes it difficult to apply Eqn. (3.1) to computing the RMF of $\mathbf{x}(t)$.

Chapter 4

Preliminaries

4.1 Definition by differential equations

First we introduce the rotation minimizing frame under weak assumptions on a spine curve, using differential equations. These results will later be connected to the classical results from differential geometry. Generally, we assume the spine curve $\mathbf{x}(u)$ to be a C^1 regular curve, i.e., $\mathbf{x}'(u) \neq 0$ in its domain of definition, but higher differentiability is needed for analysis of approximation orders. Again we use $\mathbf{t}(u) = \mathbf{x}'(u)/||\mathbf{x}'(u)||$ to denote the unit tangent vector.

Consider a one-parameter family of unit vectors $\mathbf{f}(u)$ perpendicular to the tangent vector $\mathbf{t}(u)$. Such a vector function $\mathbf{f}(u)$ is said to exhibit the *minimal rotation*, and therefore called a *rotation minimizing vector*, if it is a solution to the following system of differential–algebraic equations (DAE)

$$\left.\begin{aligned} \mathbf{f}'(u) - \phi(u)\,\mathbf{t}(u) &= 0 \\ \mathbf{f}(u) \cdot \mathbf{t}(u) &= 0 \end{aligned}\right\} \tag{4.1}$$

for the functions $\mathbf{f}(u) = (f_1(u), f_2(u), f_3(u))^\top$ and some function $\phi(u)$. Here the first equation (in vector form) constrains the evolution of $\mathbf{f}(u)$ to be parallel to the tangent, and the second equation serves to preserve orthogonality.

A rotation minimizing vector $\mathbf{f}(u)$ is not necessarily differentiable for a C^1

spine curve $\mathbf{x}(u)$; (e.g., consider the case of a C^1 curve composed of a circular arc and a straight line segment). In view of this, one may adopt the following *weak form* of the DAE (4.1)

$$\left.\begin{array}{rcl} \mathbf{f}(u) - \displaystyle\int_0^u \phi(v)\,\mathbf{t}(v)\,\mathrm{d}v &=& 0 \\ \mathbf{f}(u)\cdot\mathbf{t}(u) &=& 0 \end{array}\right\} \tag{4.2}$$

which does not involve any derivative of $\mathbf{f}(u)$.

If the spine curve is of the C^2 class, then the above DAE is equivalent to the ODE

$$\mathbf{f}'(u) = [\mathbf{t}(u)\times\mathbf{t}'(u)]\times\mathbf{f}(u) \tag{4.3}$$

since

$$\phi\,\mathbf{t} = (\mathbf{f}'\cdot\mathbf{t})\mathbf{t} = (-\mathbf{f}\cdot\mathbf{t}')\mathbf{t} = [\mathbf{t}(u)\times\mathbf{t}'(u)]\times\mathbf{f}(u) \tag{4.4}$$

A rotation minimizing frame (RMF) is determined by a rotation minimizing vector. Specifically, we have

Definition 1: [Rotation minimizing frame] *Given a C^1 curve $\mathbf{x}(u)\subset\mathbb{E}^3$, $u\in[0,L]$, a moving orthonormal frame $U(u)=(\mathbf{r}(u),\mathbf{s}(u),\mathbf{t}(u))$, where $\mathbf{r}(u)\times\mathbf{s}(u)=\mathbf{t}(u)$, is called a* rotation minimizing frame *(RMF) of $\mathbf{x}(u)$ if $\mathbf{t}(u)=\mathbf{x}'(u)/||\mathbf{x}'(u)||$ and $\mathbf{r}(u)$ is a solution of Eqn. (4.2) (or Eqn.(4.1) if $\mathbf{x}(u)$ is C^2) for some initial condition $U(0)=U_0$. Here $\mathbf{r}(u)$ is called the* reference vector *of the RMF $U(u)$.*

Since the frame vector $\mathbf{t}(u)$ of $U(u)$ is always constrained to be the unit tangent vector of $\mathbf{x}(u)$, $U(u)$ is uniquely determined by its *reference vector* $\mathbf{r}(u)$, which is a rotation minimizing vector. The third frame vector is given by $\mathbf{s}(u)=\mathbf{t}(u)\times\mathbf{r}(u)$.

The evolution defined by DAE (4.1) preserves the inner product of two vectors. Indeed, if vectors $\mathbf{f}(u)$ and $\mathbf{g}(u)$ both satisfy Eqn.(4.1) with associated functions $\phi(u)$ and $\psi(u)$, then

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{f}\cdot\mathbf{g}) = \mathbf{f}'\cdot\mathbf{g}+\mathbf{f}\cdot\mathbf{g}' = (\phi\,\mathbf{t})\cdot\mathbf{g}+\mathbf{f}\cdot(\psi\,\mathbf{t}) = 0 \tag{4.5}$$

Hence, the inner product $(\mathbf{f} \cdot \mathbf{g})$ is a constant. From this we have the following observations:

Corollary 4.1.1 *If two vectors* $\mathbf{f}_1(u)$ *and* $\mathbf{f}_2(u)$ *satisfy Eqn.* (4.1) *and the three vectors* $\mathbf{f}_1(0)$, $\mathbf{f}_2(0)$ *and* $\mathbf{t}(0)$ *form a right–handed orthonormal frame, then* $\mathbf{f}_1(u)$, $\mathbf{f}_2(u)$ *and* $\mathbf{t}(u)$ *define an RMF of the spine curve* $\mathbf{x}(u)$.

Corollary 4.1.2 *Suppose that* $\mathbf{r}(u)$ *is a rotation minimizing vector of a spine curve* $\mathbf{x}(u)$. *Then another normal vector* $\tilde{\mathbf{r}}(u)$ *of* $\mathbf{x}(u)$ *is a rotation minimizing vector of* $\mathbf{x}(u)$ *if and only if* $\tilde{\mathbf{r}}(u)$ *keeps a constant angle with* $\mathbf{r}(u)$.

Or, equivalently,

Corollary 4.1.3 *Suppose that* $U(u) = (\mathbf{r}(u), \mathbf{s}(u), \mathbf{t}(u))$ *is an RMF of a spine curve* $\mathbf{x}(u)$. *Then another right-handed orthonormal moving frame* $\tilde{U}(u) = (\tilde{\mathbf{r}}(u), \tilde{\mathbf{s}}(u), \mathbf{t}(u))$ *of* $\mathbf{x}(u)$ *is an RMF of* $\mathbf{x}(u)$ *if and only if* $\tilde{U}(u)$ *keeps a constant angle with* $U(u)$.

Finally, we note that the RMF is determined only by the geometry of a spine curve and independent of any particular parametrization $\mathbf{x}(u)$ of it.

4.2 Some differential geometry

In this subsection we shall use the arc-length parametrization $\mathbf{x}(s)$ of the spine curve. Using the Frenet formulas one may express (4.3) as

$$\mathbf{f}'(s) = \kappa(s)\mathbf{b}(s) \times \mathbf{f}(s), \tag{4.6}$$

where $\kappa(s)$ and $\mathbf{b}(s)$ are the curvature and the binormal vector of $\mathbf{x}(s)$. The vector

$$\omega_{\mathrm{RMF}}(s) = \kappa(s)\mathbf{b}(s) \tag{4.7}$$

is the angular velocity of the RMF.

The angular velocity of the Frenet frame is the so–called Darboux vector [32]

$$\omega_{\text{Frenet}}(s) = \kappa(s)\mathbf{b}(s) + \tau(s)\mathbf{t}(s) \tag{4.8}$$

This shows that, compared to the RMF, the Frenet frame involves an additional rotation around the tangent, whose speed equals the torsion τ. This observation explains the integral formula (3.1) for computing the RMF by correcting the "unwanted" rotation of the Frenet frame. The Frenet frame coincides with the RMF for planar curves, for which $\tau \equiv 0$.

The RMF is also closely related to developable surfaces and principal curvature lines of a surface. Suppose that $U(u) = (\mathbf{r}(u), \mathbf{s}(u), \mathbf{t}(u))$ is an RMF of a curve $\mathbf{x}(u)$. Then the surface $D(u, v) = \mathbf{x}(u) + v\mathbf{r}(u)$ is developable. Let $\mathbf{g}(u)$ be the edge of regression of the developable surface $D(u, v)$. Then the spine curve is an involute of the curve $\mathbf{g}(u)$. This observation suggests a natural (but restrictive) way of modeling a developable ribbon surface along a spine curve using the RMF.

Suppose that $\mathbf{x}(u)$ is a principal curvature line of a surface S. Then the consistent unit normal vector of S along the curve $\mathbf{x}(u)$ is a rotation minimizing vector of $\mathbf{x}(u)$, thus determining an RMF of $\mathbf{x}(u)$. This follows from the well known fact that the normals of S along $\mathbf{x}(u)$ form developable surface if and only if $\mathbf{x}(u)$ is a principal curvature line of S. It therefore also follows that the spine curve $\mathbf{x}(u)$ is a principal curvature line of the developable $D(u, v)$ defined in the last paragraph.

Another important property of the RMF is its preservation under conformal transformation of $\mathbb{E}^3$ [30]. This means that, given a spine curve $\mathbf{x}(u) \subset \mathbb{E}^3$ and a conformal mapping $\mathcal{C}$ of $\mathbb{E}^3$, the RMF of $\mathbf{x}(u)$ is mapped by $\mathcal{C}$ to the RMF of the transformed spine curve $\mathcal{C}(\mathbf{x}(u))$. In other words, the operation of computing

RMF of a curve and a conformal transformation commute. This property will be needed later in the analysis of the approximation order of our new method for computing the RMF.

Note that the group of conformal mappings in 3D is exactly the group generated by translations, rotations, uniform scalings and sphere inversions (reflections with respect to spheres). Since a straight line is mapped to a circle by a sphere inversion, in the above the transform of a unit vector $\mathbf{v}$ is defined by the unit tangent vector of the circle which is the image of the straight line associated with $\mathbf{v}$.

Chapter 5

Double reflection method

In this section we will first give an outline of the double reflection method, and, through a study of the RMF of a spherical curve, explain why the method works well. Then we will give a procedural description of the method that has an optimized number of arithmetic operations, and finally present an analysis of the approximation order of the method. The double reflection method is straightforward and can very easily be described; however, its justification takes interesting geometric arguments that do not appear to be trivial.

5.1 Outline of method

Given boundary data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$ and an initial right-handed orthonormal frame $U_0 = (\mathbf{r}_0, \mathbf{s}_0, \mathbf{t}_0)$ at $\mathbf{x}_0$, the next frame $U_1 = (\mathbf{r}_1, \mathbf{s}_1, \mathbf{t}_1)$ at $\mathbf{x}_1$ for RMF approximation is computed by the double reflection method in the following two steps.

Step 1 : Let $\mathcal{R}_1$ denote the reflection in the bisecting plane of the points $\mathbf{x}_0$ and $\mathbf{x}_1$ (see Figure 5.1). Use $\mathcal{R}_1$ to map U_0 to a left-handed orthonormal frame $U_0^L = (\mathbf{r}_0^L, \mathbf{s}^L, \mathbf{t}_0^L)$.

Step 2 : Let $\mathcal{R}_2$ denote the reflection in the bisecting plane of the points $\mathbf{x}_1 + \mathbf{t}_0^L$ and

$\mathbf{x}_1 + \mathbf{t}_1$ (see Figure 5.2). Use $\mathcal{R}_2$ to map U_0^L to a right-handed orthonormal frame $U_1 = (\mathbf{r}_1, \mathbf{s}_1, \mathbf{t}_1)$. Output U_1.

An efficient implementation of the above steps is given by the pseudo code in Table 5.1.

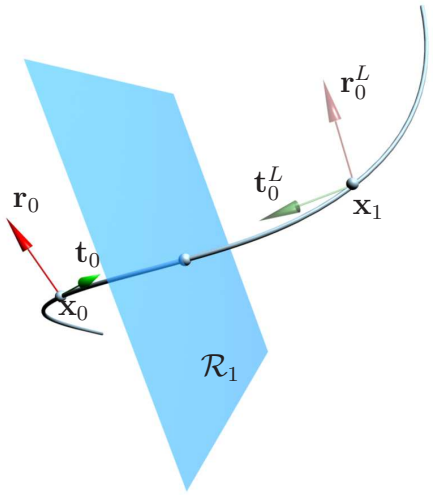


Figure 5.1: The first reflection $\mathcal{R}_1$ of the double reflection method.

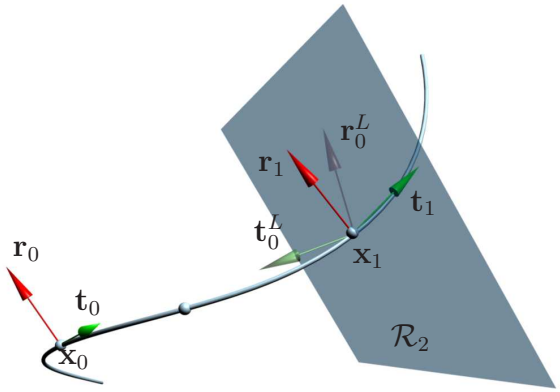


Figure 5.2: The second reflection $\mathcal{R}_2$ of the double reflection method.

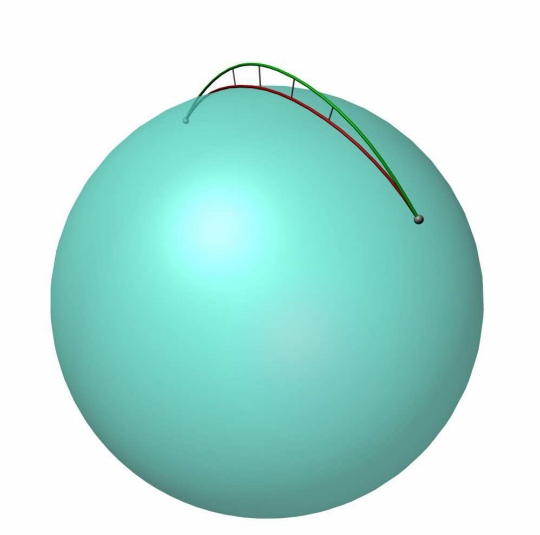


Figure 5.3: The projection of curve on sphere.

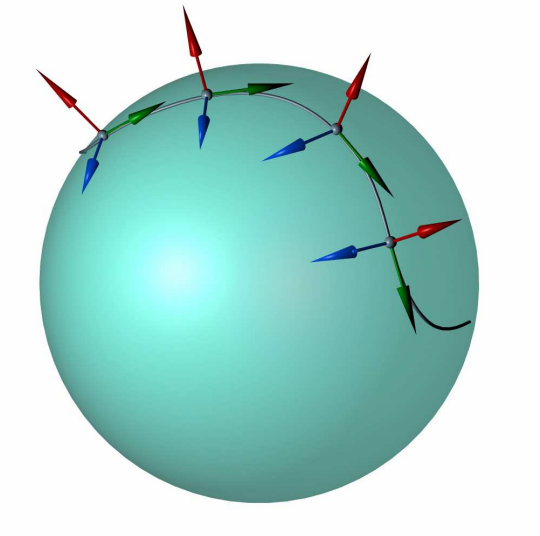


Figure 5.4: An RMF of a spherical curve.

5.2 Geometric interpretation

In the following we are going to provide an explanation for why the double reflection method described above computes an accurate approximation of an RMF, based on two key observations: 1) the double reflection method computes an exact RMF of any spherical curve; and 2) a spine curve $\mathbf{x}(u)$ with boundary data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$ is well approximated by a spherical curve $\hat{\mathbf{x}}(u)$ interpolating the same boundary data.

First consider the RMF of a spherical curve. The next lemma indicates that there is a simple explicit characterization of the RMF of a spherical curve. (We will treat a planar curve as special case of a spherical curve where the radius is infinite.)

Lemma 5.2.1 *Let $\mathbf{x}(u)$, $u \in [0, h]$, be a curve segment lying on a sphere S or a plane P (see Figure **??**). Let $\mathbf{n}(u)$ be the outward unit normal vector of the sphere S along the curve $\mathbf{x}(u)$ or a unit (constant) normal vector of the plane P. Then an RMF of $\mathbf{x}(u)$ is given by $\bar{U}_1 = (\bar{\mathbf{r}}, \bar{\mathbf{s}}, \mathbf{t}_1)$, where*

$$\bar{\mathbf{r}}(u) = \mathbf{n}(u) \ \text{ and } \ \bar{\mathbf{s}}(u) = \mathbf{t}(u) \times \mathbf{n}(u). \tag{5.1}$$

Proof. First consider the case of $\mathbf{x}(u)$ being on a sphere. Without loss of generality, suppose that the sphere S is centered at the origin and has radius r. It is clear that $\mathbf{r}(u) = \mathbf{n}(u)$, $\mathbf{s}(u) = \mathbf{t}(u) \times \mathbf{n}(u)$ and $\mathbf{t}(u)$ form a right-handed orthonormal moving frame. Since $\mathbf{n}(u) = \frac{1}{r}\mathbf{x}(u)$, $\mathbf{r}' = \mathbf{n}' = \frac{1}{r}\mathbf{x}'$, which is parallel to $\mathbf{t}(u)$. Therefore, $\mathbf{r}$ satisfies Eqn. (4.1), i.e., it is a rotational minimizing vector. Hence, by Definition 1, $U(u) = (\mathbf{r}, \mathbf{s}, \mathbf{t})$ is an RMF of $\mathbf{x}(u)$.

The proof is similar when $\mathbf{x}(u)$ is a plane curve. □.

Lemma 5.2.1 suggests that, given the initial frame U_0 at $\mathbf{x}_0$, the RMF U_1 of a

spherical curve $\mathbf{x}(u)$ at the point $\mathbf{x}_1$ does not depend on the in-between shape of $\mathbf{x}(u)$, but depends only on the boundary data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$. This will be referred to as the *path independence property*, as stated below.

Lemma 5.2.2 [Path independence property] [1] *Let $\mathbf{x}(u)$ and $\mathbf{y}(v)$ be two curve segments, $u \in [0, h_1]$ and $v \in [0, h_2]$, on a sphere (or a plane) sharing the same boundary data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$. Let $U(u)$ and $V(v)$ denote the RMFs of $\mathbf{x}(u)$ and $\mathbf{y}(v)$, having the same initial frame U_0, i.e., $U(0) = V(0) = U_0$. Then $U(h_1) = V(h_2)$.*

Proof. We will only consider the case of $\mathbf{x}(u)$ and $\mathbf{y}(u)$ being on a sphere S; the case of their being on a plane can be proved in a similar way. First suppose that the initial frame U_0 is the special frame $\bar{U}_0 = (\bar{\mathbf{r}}_0, \bar{\mathbf{s}}_0, \mathbf{t}_0)$ where $\bar{\mathbf{r}}_0$ is the unit outward normal vector of the sphere S at $\mathbf{x}_0$ and $\bar{\mathbf{s}}_0 = \mathbf{t}_0 \times \bar{\mathbf{r}}_0$. Then, by Lemma 5.2.1, the RMFs $\bar{U}_1$ and $\bar{V}_1$ of $\mathbf{x}(u)$ and $\mathbf{y}(u)$ at $\mathbf{x}_1$ are the same, i.e., $\bar{U}_1 = \bar{V}_1 = (\bar{\mathbf{r}}_1, \bar{\mathbf{s}}_1, \mathbf{t}_1)$, where $\bar{\mathbf{r}}_1$ is the unit outward normal vector $\mathbf{n}_1$ of the sphere S at $\mathbf{x}_1$ and $\bar{\mathbf{s}}_1 = \mathbf{t}_1 \times \bar{\mathbf{r}}_1$.

Now suppose that the initial frame $U_0 = (\mathbf{r}_0, \mathbf{s}_0, \mathbf{t}_0)$ is arbitrary. Let α_0 be the angle between U_0 and $\bar{U}_0$. Then, by Corollary 4.1.3, $U(h_1)$ and $\bar{U}_1$, as two RMFs of $\mathbf{x}(u)$ at the endpoint $\mathbf{x}_1$, keep the same angle α_0. Similarly, the angle between the $V(h_2)$ and $\bar{V}_1$, as two RMFs of $\mathbf{y}(v)$ at the endpoint $\mathbf{x}_1$, is also α_0. It follows that $U(h_1) = V(h_2)$, since $\bar{U}_1 = \bar{V}_1$. □.

Next we show that the double reflection method yields the exact RMF for a spherical curve.

Theorem 5.2.3 *Let $\mathbf{x}(u)$ be a curve segment, $u \in [0, h]$, on a sphere or a plane with boundary data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$. Let $U(u)$ be an RMF of $\mathbf{x}(u)$. Let $U_0 = U(0)$*

[1] This property is equivalent to the fact that the integral $\int_a^b \tau(s)\mathrm{d}s$ vanishes for closed spherical curves [32].

and $U_1 = U(h)$. Then, given boundary data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$ and the initial frame U_0, the double reflection method produces the frame U_1.

PROOF. Again we will only consider the case of the curve $\mathbf{x}(u)$ being on a sphere S; the case of a plane can be proved similarly. First consider the special case of $U_0 = \bar{U}_0 = (\bar{\mathbf{r}}_0, \bar{\mathbf{s}}_0, \mathbf{t}_0)$, as defined in the proof of Lemma 5.2.2. Then, by Lemma 5.2.1, $U_1 = \bar{U}_1 = (\bar{\mathbf{r}}_1, \bar{\mathbf{s}}_1, \mathbf{t}_1)$. Here, $\bar{\mathbf{r}}_0$ and $\bar{\mathbf{r}}_1$ are unit outward normal vectors of the sphere S at $\mathbf{x}_0$ and $\mathbf{x}_1$, respectively. Recall that in the double reflection method (cf. Section 5.1) the first reflection $\mathcal{R}_1$ is in the bisecting plane (denoted as H_1) of $\mathbf{x}_0$ and $\mathbf{x}_1$, and $\mathcal{R}_1$ maps $\bar{U}_0$ to a left-handed frame $\bar{U}_0^L = (\bar{\mathbf{r}}_0^L, \bar{\mathbf{s}}_0^L, \mathbf{t}_0^L)$. Because the two normals $\bar{\mathbf{r}}_0$ and $\bar{\mathbf{r}}_1$ of S at $\mathbf{x}_0$ and $\mathbf{x}_1$ are symmetric about the plane H_1, we have $\bar{\mathbf{r}}_0^L = \bar{\mathbf{r}}_1$.

Let H_2 denote the bisecting plane of the two points $\mathbf{x}_1 + \mathbf{t}_0^L$ and $\mathbf{x}_1 + \mathbf{t}_1$. Clearly, $\bar{\mathbf{r}}_0^L$ (or $\bar{\mathbf{r}}_1$) is contained in H_2. Since the second reflection $\mathcal{R}_2$ is in the plane H_2, it preserves $\bar{\mathbf{r}}_0^L = \bar{\mathbf{r}}_1$. Furthermore, by its construction, $\mathcal{R}_2$ maps $\mathbf{t}_0^L$ to $\mathbf{t}_1$. Therefore, $\mathcal{R}_2$ maps $\bar{U}_0^L$ to $\bar{U}_1 = (\bar{\mathbf{r}}_1, \bar{\mathbf{s}}_1, \mathbf{t}_1)$. Hence, the theorem holds in the special case of $U_0 = \bar{U}_0$.

Now consider an arbitrary initial frame $U_0 = (\mathbf{r}_0, \mathbf{s}_0, \mathbf{t}_0)$. Let α_0 denote the angle between U_0 and $\bar{U}_0$. Let $\mathcal{R}$ denote the composition of $\mathcal{R}_1$ and $\mathcal{R}_2$, i.e., the total rotation effected by the double reflection method. Clearly, $\mathcal{R}$ maps U_0 to a right-handed orthonormal frame $\hat{U}_1 = (\hat{\mathbf{r}}_1, \hat{\mathbf{s}}_1, \hat{\mathbf{t}}_1)$ such that $\hat{\mathbf{t}}_1 = \mathbf{t}_1$. Therefore, $\hat{U}_1$ and $\bar{U}_1$ differ by a rotation in the normal plane of $\mathbf{x}(u)$ at $\mathbf{x}_1$. Furthermore, since the rotation $\mathcal{R}$ is angle-preserving, the angle between $\hat{U}_1$ and $\bar{U}_1$ is also α_0, since $\mathcal{R}$ maps $\bar{U}_0$ to $\bar{U}_1$, and U_0 to $\hat{U}_1$. On the other hand, by Corollary 4.1.3, the angle between $U_1 = U(h)$ and $\bar{U}_1$ is also α_0. It follows that $\hat{U}_1 = U_1$, i.e., the exact RMF U_1 of the curve $\mathbf{x}(u)$ at $\mathbf{x}_1$ is generated by the double reflection method. □.

Not only the RMF of a spherical or plane curve $\mathbf{x}(u)$ is computed exactly by the

double reflection method, but also this computation does not refer to the sphere or the plane containing $\mathbf{x}(u)$. That is possible because of the path independence property of the RMF of a spherical curve (cf. Lemma 5.2.2). Note that when the curve segment $\mathbf{x}(u)$ is C^1 regular and parameterizes a line segment, since $\mathbf{x}(u)$ is a plane curve, its RMF is computed exactly by the double reflection method, with no need of threshold as in the projection method to avoid numerical instability.

Now consider applying the double reflection method to computing the RMF of a general spine curve $\mathbf{x}(u) \subset \mathbb{E}^3$, $u \in [0, h]$, which has boundary data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$ and is not necessarily spherical or planar. In general, there is a unique sphere S such that $\mathbf{x}_0$ and $\mathbf{x}_1$ are on S and $\mathbf{t}_0$ and $\mathbf{t}_1$ are tangent to S at $\mathbf{x}_0$ and $\mathbf{x}_1$. Let $\hat{\mathbf{x}}(u)$ denote the projection of the curve $\mathbf{x}(u)$ onto the sphere S through the center of S. Then it is easy to see that the curve $\hat{\mathbf{x}}(u)$ shares the same boundary data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$ with $\mathbf{x}(u)$ and that $\hat{\mathbf{x}}(u)$ approximates $\mathbf{x}(u)$ with an approximation error of order $\mathcal{O}(h^4)$. Since $\mathbf{x}(u)$ is well approximated by $\hat{\mathbf{x}}(u)$ and the double reflection method computes an exact RMF of the spherical curve $\hat{\mathbf{x}}(u)$, it is reasonable to believe that the double reflection method computes an accurate approximation to the RMF of the original spine curve $\mathbf{x}(u)$.

Note that the above argument does not constitute a formal analysis of the approximation accuracy of the double reflection method; it merely provides a geometric and intuitive understanding of why the method is expected to work well for RMF computation. It will be proved in Section 5.6 that the global approximation error of the double reflection method has the order $\mathcal{O}(h^4)$.

5.3 Procedural description

The description of the double reflection method in Section 5.1, though simple in geometric terms, is not for efficient implementation. In this section we will

give a procedural description of the method, aiming at minimizing the number of arithmetic operations required.

Since only transformation of vectors matters in RMF computation, we may just use the linear parts, denoted by matrices R_1 and R_2, of the two reflections $\mathcal{R}_1$ and $\mathcal{R}_2$. Since $\mathcal{R}_1$ is a reflection in a plane with normal vector $\mathbf{v}_1 \equiv \mathbf{x}_1 - \mathbf{x}_0$, it can be shown that its linear part is

$$R_1 = I - 2(\mathbf{v}_1\mathbf{v}_1^T)/(\mathbf{v}_1^T\mathbf{v}_1), \tag{5.2}$$

where I is the 3×3 identity matrix. We will call $\mathbf{v}_1$ the *reflection vector* of R_1. (Note that R_1 is none other than the Householder transform used for QR matrix decomposition.)

The reflection $\mathcal{R}_2$ has the reflection vector $\mathbf{v}_2 \equiv (\mathbf{x}_1+\mathbf{t}_1)-(\mathbf{x}_1+\mathbf{t}_0^L) = \mathbf{t}_1-\mathbf{t}_0^L$. So its linear part is

$$R_2 = I - 2(\mathbf{v}_2\mathbf{v}_2^T)/(\mathbf{v}_2^T\mathbf{v}_2). \tag{5.3}$$

Let $\mathbf{r}_0$ be the reference vector of U_0. Then $\mathbf{r}_1 = R_2R_1\mathbf{r}_0$ is the reference vector $\mathbf{r}_1$ of the next frame U_1. With the known tangent vector $\mathbf{t}_1$, the remaining vector $\mathbf{s}_1$ of $U_1 = (\mathbf{r}_1, \mathbf{s}_1, \mathbf{t}_1)$ is given by $\mathbf{s}_1 = \mathbf{t}_1 \times \mathbf{r}_1$.

The procedure of the double reflection method is given in Table 5.1. For a given sequence of sampled points $\mathbf{x}_i$ and associated unit tangent vectors $\mathbf{t}_i$, with an initial frame U_0 defined at $\mathbf{x}_0$, one just needs to apply the two reflections R_1 and R_2 to successively generate the approximate RMF U_i at $\mathbf{x}_i$. In each step, from the current frame U_i, we form the first reflection R_1 following Eqn.(5.2) and use R_1 to map the reference vector $\mathbf{r}_i$ to $\mathbf{r}_i^L$, and also the tangent vector $\mathbf{t}_i$ to $\mathbf{t}_i^L$. Then we use $\mathbf{t}_i^L$ and $\mathbf{t}_{i+1}$ to form the second reflection R_2 following Eqn. (5.3) and use R_2 to map $\mathbf{r}_i^L$ to the reference vector $\mathbf{r}_{i+1}$ of the next frame U_{i+1}.

Table 5.1: **Algorithm — *Double Reflection***

Input: Points $\mathbf{x}_i$ and associated unit tangent vectors $\mathbf{t}_i$, $i = 0, 1, \ldots, n$.
An initial frame $U_0 = (\mathbf{r}_0, \mathbf{s}_0, \mathbf{t}_0)$.
Output: $U_i = (\mathbf{r}_i, \mathbf{s}_i, \mathbf{t}_i)$, $i = 0, 1, 2, \ldots, n$, as approximate RMF.

```
Begin
for i = 0 to n − 1 do
      Begin
1)     v1 := xi+1 − xi;              /*compute reflection vector of R1.
*/
2)     c1 := v1 · v1;
3)     riL := ri − (2/c1) ∗ (v1 · ri) ∗ v1;      /*compute riL = R1ri.   */
4)     tiL := ti − (2/c1) ∗ (v1 · ti) ∗ v1;       /*compute tiL = R1ti.   */
5)     v2 := ti+1 − tiL;             /*compute reflection vector of R2.
*/
6)     c2 := v2 · v2;
7)     ri+1 := riL − (2/c2) ∗ (v2 · riL) ∗ v2;    /*compute ri+1 = R2riL.   */
8)     si+1 := ti+1 × ri+1;          /*compute vector si+1 of Ui+1.   */
9)     Ui+1 := (ri+1, si+1, ti+1);
      End
End
```

5.4 Degenerate cases and symmetry

By degeneracy we mean that either of the reflections $\mathcal{R}_1$ and $\mathcal{R}_2$ becomes undefined. Clearly, $\mathcal{R}_1$ is undefined if and only if $\mathbf{x}_1 - \mathbf{x}_0 = 0$, and $\mathcal{R}_2$ is undefined if and only if $\mathbf{x}_1 + \mathbf{t}_0^L = \mathbf{x}_1 + \mathbf{t}_1$, i.e., the two points $\mathbf{x}_0 + \mathbf{t}_0$ and $\mathbf{x}_1 + \mathbf{t}_1$ are symmetric about the bisecting plane of $\mathbf{x}_0$ and $\mathbf{x}_1$; this is equivalent to $(\mathbf{x}_1 - \mathbf{x}_0) \cdot (\mathbf{t}_1 + \mathbf{t}_0) = 0$ and $(\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{t}_1 - \mathbf{t}_0) = 0$. Hence, for proper application of the double reflection method, we need to ensure that the following two conditions are satisfied: (1) $\mathbf{x}_1 - \mathbf{x}_0 \neq 0$; and (2) $(\mathbf{x}_1 - \mathbf{x}_0) \cdot (\mathbf{t}_1 + \mathbf{t}_0) \neq 0$ or $(\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{t}_1 - \mathbf{t}_0) \neq 0$. Both conditions are simple to test and can easily be satisfied provided that the spine curve is sufficiently subdivided or sampled.

The double reflection method is symmetric in the following sense. Given a sequence of sampled points $\mathbf{x}_i$, $i = 0, 1, \ldots, n$, on a spine curve $\mathbf{x}(u)$, suppose that the U_i are the frames computed by the double reflection method applied to $\mathbf{x}(u)$ with U_0 as the initial frame. Then the same sequence of frames in the reversed order, i.e., U_{n-i}, $i = 0, 1, \ldots, n$, will be generated by applying the double reflection method starting from $\mathbf{x}_n$, using U_n as the initial frame. This symmetry property can be proved by examining the basic steps of the double reflection method, but we will skip the proof. The projection method and the rotation method also possess this symmetry property, while the Runge–Kutta method does not.

5.5 Invariance under conformal mappings

We have seen that conformal mappings in 3D preserve the RMF of a space curve (cf. Section 4.2). It turns out that the approximate RMF computed with the double reflection method is also preserved by conformal mappings, in the following sense. Suppose that the sampled points $\mathbf{x}_i$ of a spine curve $\mathbf{x}(u)$ are used to com-

pute the approximate RMF U_i of $\mathbf{x}(u)$. Then the images of U_i under a conformal mapping $\mathcal{C}$ are the same as the approximate RMF of the curve $\mathcal{C}(\mathbf{x}(u))$ that are computed by the double reflection method using the sampled points $\mathcal{C}(\mathbf{x}_i)$.

This property follows easily from the fact that the basic step of the double reflection method is performed on the sphere S_i touching the two ends of the data $(\mathbf{x}_i, \mathbf{t}_i; \mathbf{x}_{i+1}, \mathbf{t}_{i+1})$ and this sphere is preserved by any conformal mapping $\mathcal{C}$ (which are a sequence of sphere inversions), i.e., the image $\mathcal{C}(S_i)$ is the sphere touching the transformed data $(\mathcal{C}(\mathbf{x}_i), \mathcal{C}(\mathbf{t}_i); \mathcal{C}(\mathbf{x}_{i+1}), \mathcal{C}(\mathbf{t}_{i+1}))$.

Since both exact RMF and approximate RMF computed with the double reflection method are preserved by conformal mappings, and the conformal mapping is angle preserving, we conclude that the approximation error of the double reflection method is invariant under conformal mappings.

The double reflection method is an ideal method from the viewpoint of discrete differential geometry. Because the exact RMF of a smooth curve is preserved by conformal mappings, we naturally expect that a good method acting on a discretization of the curve for computing its approximate RMF is invariant under the same group of transformations. The double reflection method indeed satisfies this property. We note that the projection method, the rotation method and the Runge–Kutta method do not possess this property.

5.6 Order of approximation

First consider an analytic curve segment with the arc length parametrization $\mathbf{x}(s)$, $s \in [0, h]$, of length h. Suppose that the initial frame $U(0) = U_0 \equiv (\mathbf{r}_0, \mathbf{s}_0, \mathbf{y}_0)$ of an RMF $U(s)$ of $\mathbf{x}(s)$ is given. We approximate the frame $U(h)$ at $\mathbf{x}_1 = \mathbf{x}(h)$ by the frame U_1 computed with the the double reflection method.

Theorem 5.6.1 *The one-step error $U(h) - U_1$ in RMF computation introduced by the double reflection method has the order of $\mathcal{O}(h^5)$. Specifically,*

$$\|\mathbf{r}(h) - \mathbf{r}_1\| = \tfrac{1}{720} K h^5 + \mathcal{O}(h^6). \tag{5.4}$$

Here $K = 2\,{\kappa_1}^2\tau_0 + {\kappa_0}^2{\tau_0}^3 + \kappa_1\kappa_0\tau_1 - \kappa_2\kappa_0\tau_0$ is a bounded constant for a smooth curve, where $\kappa_i = (\mathrm{d}/\mathrm{d}s)^i\kappa(s)|_{s=0}$, $\tau_i = (\mathrm{d}/\mathrm{d}s)^i\tau(s)|_{s=0}$ are the curvature, torsion and their respective derivatives at $s = 0$.

The proof of Theorem 5.6.1 is given in next section. The constant K in Eqn.(5.4) has an interesting geometric interpretation. A spherical curve $\mathbf{x}(s)$ is characterized by the differential equation [32]

$$\frac{\tau}{\kappa} - \frac{\mathrm{d}}{\mathrm{d}s}\left\{\frac{\kappa'}{\kappa^2\tau}\right\} = 0.$$

It is easy to verify that the numerator of this equation is

$$K(s) = 2\,{\kappa_1(s)}^2\tau_0(s) + {\kappa_0(s)}^2{\tau_0(s)}^3 + \kappa_1(s)\kappa_0(s)\tau_1(s) - \kappa_2(s)\kappa_0(s)\tau_0(s).$$

Therefore, $K(s) = 0$ if and only if $\mathbf{x}(s)$ is a spherical curve. Hence, intuitively speaking, $K = K(0)$ measures how close $\mathbf{x}(s)$ is to a spherical curve at $s = 0$.

As an obvious corollary of Theorem 5.6.1, we have the next theorem that the RMF computation by the double reflection method applied to a general regularly parameterized spine curve has the fourth order global approximation error.

Theorem 5.6.2 *Given a regularly parameterized spine curve $\mathbf{x}(u)$, $u \in [0, M]$, let $\mathbf{x}_i = \mathbf{x}(u_i)$, $i = 0, 1, \ldots, n$, be points sampled on $\mathbf{x}(u)$ with equally spaced parameter values, i.e., $u_i = i * h$ and $h = M/n$. Then the global error of the approximate RMF of $\mathbf{x}(u)$ computed by the double reflection method applied to the sequence $\{\mathbf{x}_i\}$ has the order $\mathcal{O}(h^4)$.*

5.7 Proof of Theorem 5.6.1

There are two parts in this proof. In the first part we derive an expression of the order $\mathcal{O}(h^5)$ term of the one-step error. In the second part we show that coefficient of this error term is bounded for a regular curve, thus yielding the claimed order of magnitude.

We will obtain the error expression using the canonical Taylor expansion of the curve $\mathbf{x}(s)$ at $\mathbf{x}(0)$, which can be derived from the Frenet formulas [32]. In a neighborhood of $\mathbf{x}(0)$, $\mathbf{x}(s)$ is approximated by the series

$$\mathbf{x}(s) = \begin{pmatrix} s & -\frac{1}{6}\kappa_0^2 s^3 & -\frac{1}{8}\kappa_0\kappa_1 s^4 & +\cdots \\ \frac{1}{2}\kappa_0 s^2 & +\frac{1}{6}\kappa_1 s^3 & +\frac{1}{24}(\kappa_2 - \kappa_0^3 - \tau_0^2\kappa_0)s^4 & +\cdots \\ & +\frac{1}{6}\kappa_0\tau_0 s^3 & +\frac{1}{24}(\kappa_0\tau_1 + 2\kappa_1\tau_0)s^4 & +\cdots \end{pmatrix}, \tag{5.5}$$

where the Frenet frame at $s = 0$ is aligned with the axes of the Cartesian coordinates, and $\kappa_i = (\mathrm{d}/\mathrm{d}s)^i\kappa(s)|_{s=0}$, $\tau_i = (\mathrm{d}/\mathrm{d}s)^i\tau(s)|_{s=0}$. With the help of computer algebra tools, we generate Taylor series for all quantities needed for computing the variables listed in the procedure of the double reflection method (Table 5.1). Due to space limitation, only an outline of the derivation will be given.

Consider a segment of $\mathbf{x}(s)$ of length h starting at the origin, i.e.,

$$(0,0,0)^\top = \mathbf{x}_0 = \mathbf{x}(0),\ \mathbf{x}_1 = \mathbf{x}(h),\ (1,0,0)^\top = \mathbf{t}_0 = \dot{\mathbf{x}}(0),\ \mathbf{t}_1 = \dot{\mathbf{x}}(h). \tag{5.6}$$

Let $\mathbf{r}_0 = (0, C, S)$, where $C^2 + S^2 = 1$, be the reference vector of U_0 at $\mathbf{x}_0$. We compute the new reference vector $\mathbf{r}_1$ using steps from (1) to (7) of the algorithm

Double Reflection (see Table 5.1):

$$\mathbf{v}_1 = (h + \mathcal{O}(h^3),\ \tfrac{1}{2}\kappa_0 h^2 + \mathcal{O}(h^3),\ \mathcal{O}(h^3))^\top$$

$$c_1 = h^2 - \tfrac{1}{12}{\kappa_0}^2 h^4 + \mathcal{O}(h^5)$$

$$\mathbf{r}_0^L = (-C\kappa_0 h - \tfrac{1}{3}(C\kappa_1 + \kappa_0\tau_0 S)h^2 + \mathcal{O}(h^3),\ C - \tfrac{1}{2}{\kappa_0}^2 C h^2 + \mathcal{O}(h^3), S + \mathcal{O}(h^3))^\top$$

$$\mathbf{t}_0^L = (-1 + \tfrac{1}{2}{\kappa_0}^2 h^2 + \mathcal{O}(h^3), -\kappa_0 h - \tfrac{1}{3}\kappa_1 h^2 + \mathcal{O}(h^3), -\tfrac{1}{3}\kappa_0\tau_0 h^2 + \mathcal{O}(h^3))^\top$$

$$\mathbf{v}_2 = (2 - {\kappa_0}^2 h^2 + \mathcal{O}(h^3), 2\kappa_0 h + \tfrac{5}{6}\kappa_1 h^2 + \mathcal{O}(h^3), \tfrac{5}{6}\kappa_0\tau_0 h^2 + \mathcal{O}(h^3))^\top$$

$$c_2 = 4 - \tfrac{1}{36}({\tau_0}^2{\kappa_0}^2 + {\kappa_1}^2)h^4 + \mathcal{O}(h^5)$$

$$\mathbf{r}_1 = (-C\kappa_0 h - \tfrac{1}{2}(C\kappa_1 + \kappa_0\tau_0 S)h^2 + \mathcal{O}(h^3), C - \tfrac{1}{2}{\kappa_0}^2 C h^2 + \mathcal{O}(h^3), S + \mathcal{O}(h^3))^\top$$

On the other hand, using the angular velocity of the RMF (Eqn. (4.7)) we generate the Taylor expansion of the reference vector $\mathbf{r}(h)$ of the exact RMF $U(h)$,

$$\mathbf{r}(h) = \mathbf{r}(s)\Big|_{s=0} + \underbrace{\kappa(s)\mathbf{b}(s)\times\mathbf{r}(s)}_{=\mathbf{r}'(0)}\Big|_{s=0} h + \underbrace{\frac{\mathrm{d}}{\mathrm{d}s}(\kappa(s)\mathbf{b}(s)\times\mathbf{r}(s))}_{=\mathbf{r}''(0)}\Big|_{s=0}\frac{h^2}{2} + \dots$$

Using the Frenet formulas and the fact that the derivatives of $\mathbf{r}(s)$ are given by the previously generated terms of the Taylor expansion, $\mathbf{r}(h)$ can be expressed solely by using derivatives of curvature and torsion at $s = 0$, and by the initial value $\mathbf{r}(0) = (0, C, S)^\top$. Finally, we compare the Taylor expansions of $\mathbf{r}(h)$ and $\mathbf{r}_1$ to obtain

$$\mathbf{r}(h) - \mathbf{r}_1 = (\mathcal{O}(h^6), -\tfrac{1}{720}\,S\,K\,h^5 + \mathcal{O}(h^6), \tfrac{1}{720}\,C\,K\,h^5 + \mathcal{O}(h^6))^\top,$$

where

$$K = 2\,{\kappa_1}^2\tau_0 + {\kappa_0}^2{\tau_0}^3 + \kappa_1\kappa_0\tau_1 - \kappa_2\kappa_0\tau_0 \tag{5.7}$$

Hence,

$$\|\mathbf{r}(h) - \mathbf{r}_1\| = \frac{1}{720}Kh^5 + \mathcal{O}(h^6)$$

Next, we need to show that the coefficient K in the $\mathcal{O}(h^5)$ term above is finite for a regular smooth curve. This is a concern because the torsion τ_0 appearing

in K (Eqn. (5.7)) and τ_0 can become unbounded for a regular curve; such an example is given in Appendix II. Note that only the curvature κ_0, torsion τ_0 and their derivatives are present in K. Sincec

$$\kappa(s) = \|\ddot{\mathbf{x}}(s)\|, \quad \tau(0) = \frac{\|(\dot{\mathbf{x}}(s) \times \ddot{\mathbf{x}}(s)) \cdot \dddot{\mathbf{x}}(s)\|}{\|\ddot{\mathbf{x}}(s)\|^3}$$

it is easy to see that, if a spine curve has non-vanishing curvature, then $\kappa_0 = \kappa(0)$ is bounded from zero, and $\tau_0 = \tau(0)$ and its derivative are finite; consequently, K will be finite in this case.

We will use a conformal mapping to turn an arbitrary curve segment $\mathbf{x}(s)$, $s \in [0, h]$, possibly with vanishing curvature, into another curve segment with curvature bounded from zero. First take the osculating plane of $\mathbf{x}(s)$ at $s = 0$. With a rigid motion we take this plane to be the x-y plane and have the point $\mathbf{x}(0)$ positioned at the origin $(0, 0, 0)$. Let $\mathcal{C}_s$ denote the inversion with respect to the sphere S_1 of radius 1 and centered at $(0, 0, 1)$. Then the plane x-y is mapped by $\mathcal{C}_s$ to the sphere S_2 of radius $1/2$ and centered at $(0, 0, 1/2)$. Clearly, $\mathcal{C}_s$ is conformal, and the point $\mathbf{x}(0) = (0, 0, 0)$ is fixed by $\mathcal{C}_s$.

Let κ_0 be the curvature of $\mathbf{x}(s)$ at $s = 0$. Let $\mathbf{x}_c(s)$ denote the transformed curve $\mathcal{C}_s(\mathbf{x}(s))$. With a bit abuse of notation, we use $\mathbf{x}_c(t)$, $t \in [0, h_c]$, to denote arclength parametrization of the segment $\mathbf{x}_c(s)$. At $t = 0$, the curve $\mathbf{x}_c(t)$ has the normal curvature equal to 2, which is the reciprocal of the radius of S_2, and the geodesic curvature equal to κ_0, which is the curvature of $\mathbf{x}(s)$ at $s = 0$. (The curve $\mathbf{x}_c(s)$ has the same normal curvature and geodesic curvature at $\mathbf{x}_c(0)$ as any spherical curve on S_2 that has the second order contact with $\mathbf{x}_c(s)$ at $\mathbf{x}_c(0)$.) It follows that the curvature of $\mathbf{x}_c(u)$ at $\mathbf{x}_c(0)$ is $\kappa_c = (\kappa_0^2 + 4)^{1/2}$.

Clearly, κ_c is bounded away from zero. Hence, if we apply the double reflection method to the transformed curve segment $\mathbf{x}_c(t)$, $t \in [0, h_c]$, according to the preceding analysis, the fifth order term of the approximation error takes the form

$\frac{1}{720}K_c h_c^5$; here K_c is finite, since κ_c is bounded away from zero. On the other hand, because the approximation error produced by the double reflection method is invariant under a conformal mapping (cf. Section 5.5), in the limit we have

$$\frac{K}{720}h^5 = \frac{K_c}{720}h_c^5$$

When h is sufficiently small, due to the regular nature of the mapping $\mathcal{C}_s$ in the neighborhood of $\mathbf{x}(0)$, there exists a constant $d > 0$ such that $h_c < dh$. It follows that

$$K = \frac{h_c^5}{h^5}K_c < d^5 K_c$$

Hence, K is also finite. This completes the proof that the local one-step error of the double reflection method is of the order of $\mathcal{O}(h^5)$. □

Chapter 6

Comparison and Experiments

6.1 Computational cost

We need to count the numbers of operations in order to compare the efficiency of different methods for RMF computation. First consider the operation cost of the double reflection method for computing each new frame U_{i+1} from U_i, following the procedure in Table 5.1. We will count a subtraction as equivalent to an addition. Step (1) uses 3 *adds*. Step (2) uses 2 *adds*, 3 *mults*. After evaluating c_1 with 1 *div*, step (3) can be completed using 5 *adds*, 7 *mults* and 1 *div*. Similarly, step (4) can be done using, 5 *adds*, 7 *mults* and 1 *div*. Step (5) uses only 3 *adds*. Step (6) uses 2 *adds*, and 3 *mults*. Step (7) uses 5 *adds*, 6 *mults* and 1 *div*. Finally, step (8) uses 3 *adds* and 6 *mults*. Hence, in total, the per frame computation of the double reflection method costs, 28 additions, 32 multiplications and 2 divisions.

As comparison, we next give the operation counts of the projection method and the rotation method. In the projection method [31], the new reference vector $\mathbf{r}_1$ can be computed from $\mathbf{r}_0$ by

$$\mathbf{r}_1 = \mathbf{r}_0 - \frac{\mathbf{r}_0 \cdot \mathbf{t}_1}{(\mathbf{x}_1 - \mathbf{x}_0) \cdot \mathbf{t}_1}(\mathbf{x}_1 - \mathbf{x}_0).$$

This evaluation takes 9 *mults* and 1 *div*. Since $\mathbf{r}_1$ thus derived is in general not a unit vector, 6 *mults*, 1 *div* and one square root are needed to normalize $\mathbf{r}_1$. Then

another 6 *mults* are needed to compute the third frame vector $\mathbf{s}_1 = \mathbf{t}_1 \times \mathbf{r}_1$. Hence, in total, the projection method needs 15 additions, 21 multiplications, 2 divisions and 1 square root to compute a new frame. This is less than, but comparable to, the cost of the double reflection method.

A procedure of the rotation method is given in [40]. Given the two consecutive unit tangent vectors $\mathbf{t}_0$ and $\mathbf{t}_1$, the rotation axis is computed as $(a, b, c) = \mathbf{t}_0 \times \mathbf{t}_1$ and the cosine of rotation angle is $\cos\alpha = \mathbf{t}_0 \cdot \mathbf{t}_1$. Then the rotation matrix is given by

$$R = \begin{bmatrix} \cos\alpha & -c & b \\ c & \cos\alpha & -a \\ -b & a & \cos\alpha \end{bmatrix} + \frac{1-\cos\alpha}{a^2+b^2+c^2} \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 & bc \\ ac & bc & c^2 \end{bmatrix}.$$

Therefore, 21 *mults* and 1 *div* are needed to obtain R from $\mathbf{t}_0$ and $\mathbf{t}_1$; (note that 27 *mults* are claimed in [40]). In addition, 9 *mults* are needed for computing the next reference vector $\mathbf{r}_1 = R\mathbf{r}_0$, and 6 *mults* for computing the remaining vector $\mathbf{s}_1 = \mathbf{t}_1 \times \mathbf{r}_1$. Hence, in total, the rotation method needs 26 additions, 36 multiplications and 1 division.

The number of operations for the three methods are listed in Table 6.1. The three methods have similar computational costs, as our tests show that a *sqrt* or a division is is about six times more time consuming than a multiplication. The actual timing comparison will be given in the next subsection.

It is worth mentioning that another procedure of the rotation method is given in [9], which uses 19 *mults* and a square root to compute the rotation matrix R after using 6 *mults* to get the rotation axis $\mathbf{t}_0 \times \mathbf{t}_1$. Hence, that version of the rotation method requires in total 40 multiplications and a square root to compute a new frame, assuming that the $\mathbf{t}_i$ are unit tangent vectors. In the subsequent experimental comparisons involving the rotation method we will refer to the faster implementation in [40].

Method	# of *adds*	# of *mults*	# of *divs*	# of *sqrt*
Projection	15	21	2	1
Rotation	26	36	1	0
Double reflection	28	32	2	0

Table 6.1: The operations counts of the three methods.

6.2 Experimental results

We will use two examples to compare the double reflection method with the following existing methods: the projection method, the rotation method and the 4-th order Runge-Kutta method, in terms of efficiency and accuracy. All test cases were run on a PC with Intel Xeon 2.66 GHz CPU and 2.00 GB RAM.

Example 1: In the first example we use the four methods to compute the RMF of the spine curve, which is a torus knot, given by

$$\mathbf{x}(u) = [(0.6{+}0.3\cos(7u))\cos(2u), (0.6{+}0.3\cos(7u))\sin(2u), 0.3\sin(7u)]^T, \quad u \in [0, L] \tag{6.1}$$

We compute the RMF using different step sizes $h = 0.01 * 2^{-k}$, $k = 0, 1, \ldots$; that is, for each fixed step size h, the sampled points are $\mathbf{x}(i * h)$, $i = 0, 1, \ldots, L/h$.

The timings of computing the sequence of frames by the four methods are shown in Figures 6.1 and 6.2. We see that the projection method, the rotation method and the double reflection method have similar time costs. The Runge-Kutta method costs much more time than the double reflection method, since it needs more function evaluations in each step than the other three methods.

To observe approximation errors, we need an exact RMF of the spine curve or an approximate RMF of very high accuracy against which the computed approximate RMF by the four methods can be compared. Since the exact RMF of the torus knot given by Eqn.(6.1) is difficult to obtain, we use the integration function provided in Maple to get an approximate RMF of $\mathbf{x}(u)$ whose approximation error

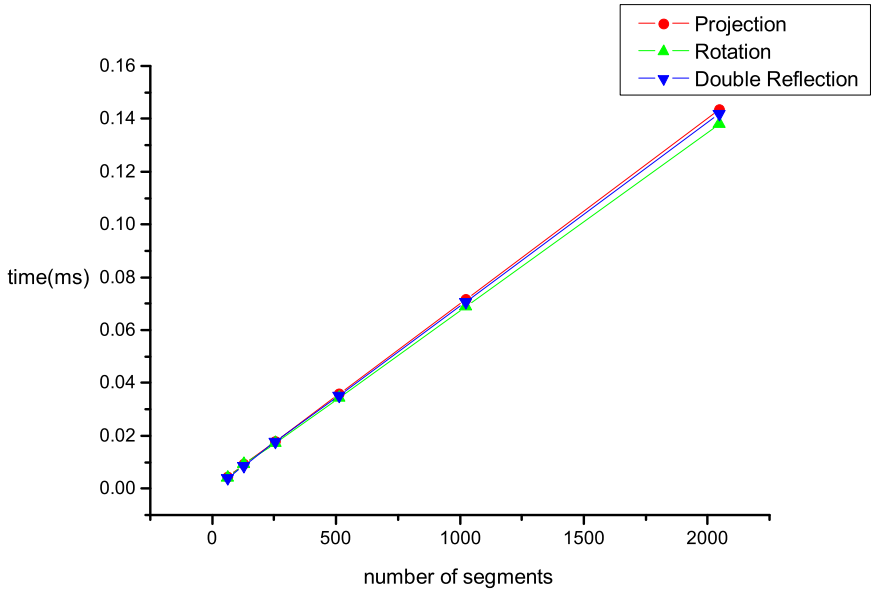


Figure 6.1: Timings of the double reflection method, the projection method and the rotation method.

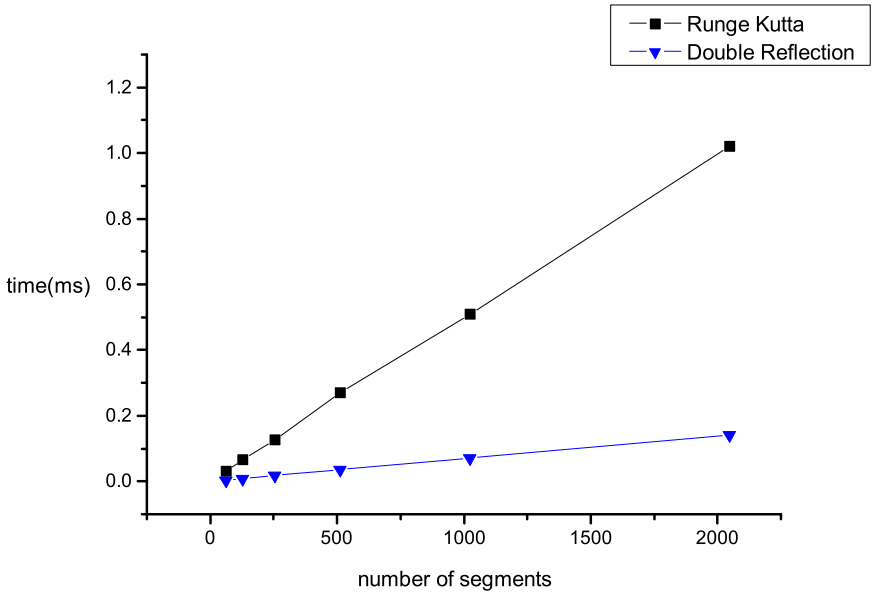


Figure 6.2: Timings of Runge-Kutta method and the double reflection method.

is known to be less than 10^{-16}. This highly accurate RMF is used in place of an exact RMF to measure the global approximation error E_g defined in (1.1).

The global approximation errors e_k of the four methods are shown in Figure 6.3 and also in Tables 6.2 and 6.3, where e_k is the error of using 2^k segments, $k = 6, 7, \ldots, 11$. These data confirm that the projection method and the rotation method have the second order of global approximation error $\mathcal{O}(h^2)$, and the Runge-Kutta method and the double reflection method have the fourth order of global approximation error $\mathcal{O}(h^4)$. We use the computed sequences of frames by the four methods to generate ribbon-like sweep surfaces with the torus knot $\mathbf{x}(u)$ as the spine curve, and show the four surfaces in Figures 6.5 through 6.8 using color coding to indicate the magnitude of the approximation errors.

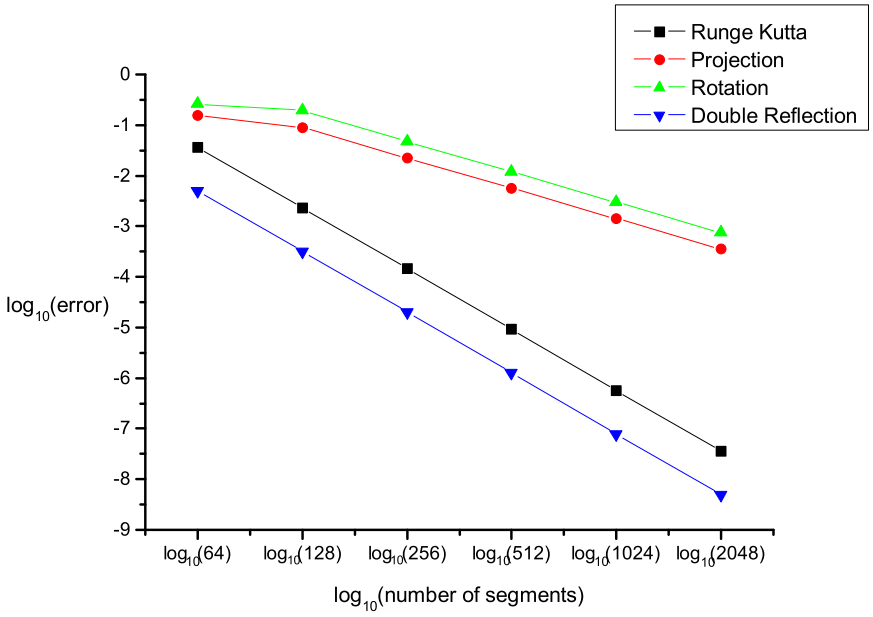


Figure 6.3: Global errors of the four methods for the torus knot in Example 1.

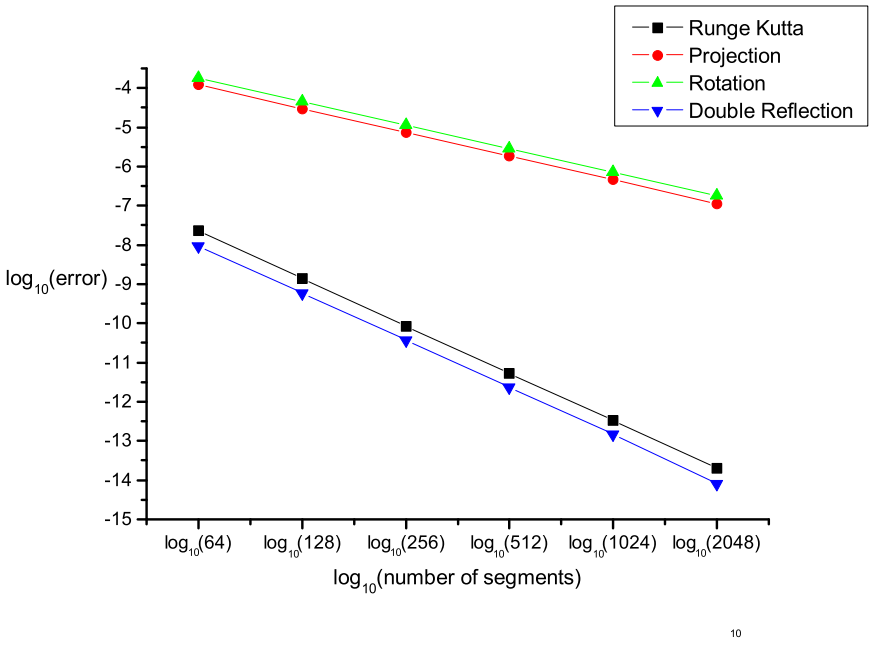


Figure 6.4: Global errors of the four methods for the PH curve in Example 2.

	Double reflection	Runge-Kutta
# of segments	error e_k, ratio e_k/e_{k-1}	error e_k, ratio e_k/e_{k-1}
2^6	5.10E−3, N.A.	3.58E−2, N.A.
2^7	3.24E−4, 0.063577	2.32E−3, 0.064846
2^8	2.03E−5, 0.062776	1.46E−4, 0.062737
2^9	1.27E−6, 0.062571	9.10E−6, 0.062408
2^{10}	7.95E−8, 0.062578	5.68E−7, 0.062422
2^{11}	4.97E−9, 0.062575	3.55E−8, 0.062438

Table 6.2: Global approximation errors e_k of the double reflection method and by using the 4-th order Runge-Kutta method for the torus knot in Example 1. The error ratios e_k/e_{k-1} show that the approximation orders of these two methods are both $\mathcal{O}(h^4)$.

	Projection method	Rotation method
# of segments	error e_k, ratio e_k/e_{k-1}	error e_k, ratio e_k/e_{k-1}
2^6	1.56E−1, N.A.	2.60E−1, N.A.
2^7	9.03E−2, 0.579295	1.91E−1, 0.736606
2^8	2.26E−2, 0.249757	4.76E−2, 0.248776
2^9	5.64E−3, 0.249939	1.19E−2, 0.249668
2^{10}	1.41E−3, 0.249983	2.97E−3, 0.249906
2^{11}	3.52E−4, 0.249995	7.42E−4, 0.249971

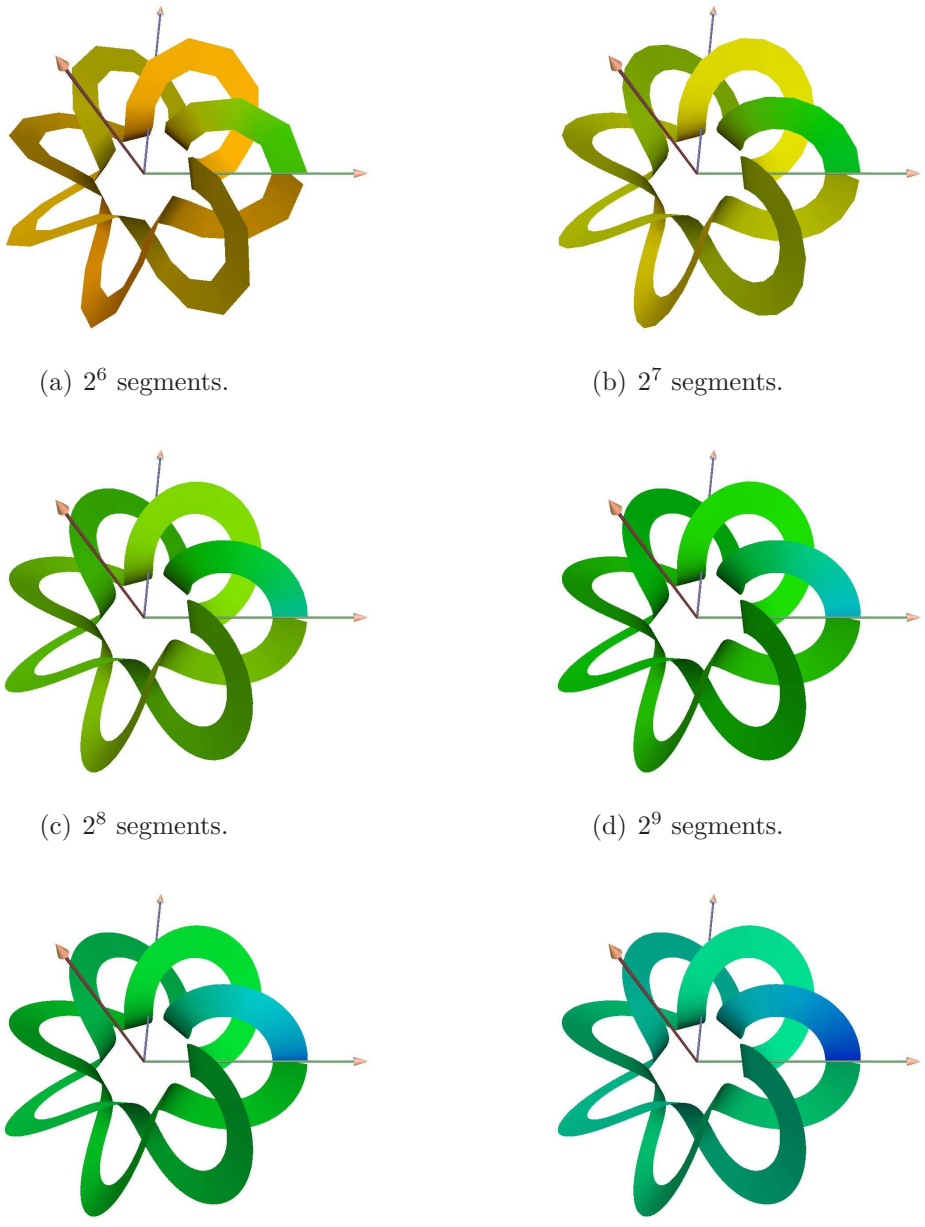
Table 6.3: Global approximation errors e_k of the projection method and the rotation method for the torus knot in Example 1. The error ratios e_k/e_{k-1} show that the approximation orders of these two methods are both $\mathcal{O}(h^2)$.

Example 2: In the second example we use the double reflection method to approximate the RMF of a PH (Pythagorean-hodograph) curve, whose RMF can be computed exactly by a closed-form formula [18]. Given two points $\mathbf{x}_0 = (1000, 0, 0)^T$ and $\mathbf{x}_1 = (1000, 2000, 4000)^T$ with associated un-normalized tangent vectors $\hat{\mathbf{t}}_0 = (1, 5, -1)^T, \hat{\mathbf{t}}_1 = (-3, 2, 5)^T$, we obtain a cubic PH curve $\mathbf{x}(u)$ as the spine curve using G^1 Hermite interpolation, following [29]. Let the Frenet frame of $\mathbf{x}(u)$ at $u = 0$ be the initial frame U_0. Compared with the exact RMF of $\mathbf{x}(u)$ at the endpoint $\mathbf{x}_1 = \mathbf{x}(1)$, we obtain the errors of the approximate RMF computed by the four methods. These errors are shown in Figure 6.4. The errors of the double reflection method and the rotation method are also given in Table 6.4 and

	Double reflection	Rotation method
# of segments	error e_k, ratio e_k/e_{k-1}	error e_k, ratio e_k/e_{k-1}
2^6	9.29E−9, N.A.	1.78E−4, N.A.
2^7	5.94E−10, 0.063919	4.47E−5, 0.250721
2^8	3.75E−11, 0.063181	1.12E−5, 0.250321
2^9	2.36E−12, 0.062926	2.80E−6, 0.250151
2^{10}	1.48E−13, 0.062789	7.00E−7, 0.250073
2^{11}	9.25E−15, 0.062521	1.75E−7, 0.250036

Table 6.4: Global approximation errors e_k of the double reflection method and the rotation method for the PH curve. The error ratios e_k/e_{k-1} confirm again the $\mathcal{O}(h^4)$ approximation order of the double reflection method and the $\mathcal{O}(h^2)$ approximation order of the rotation method.

their color coded surface representations in Figure 6.9. These data confirm again the fourth order approximation error $\mathcal{O}(h^4)$ of the double reflection method.



(a) 2^6 segments.

(b) 2^7 segments.

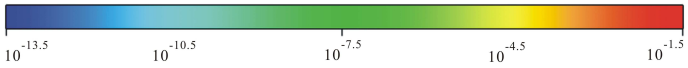
(c) 2^8 segments.

(d) 2^9 segments.

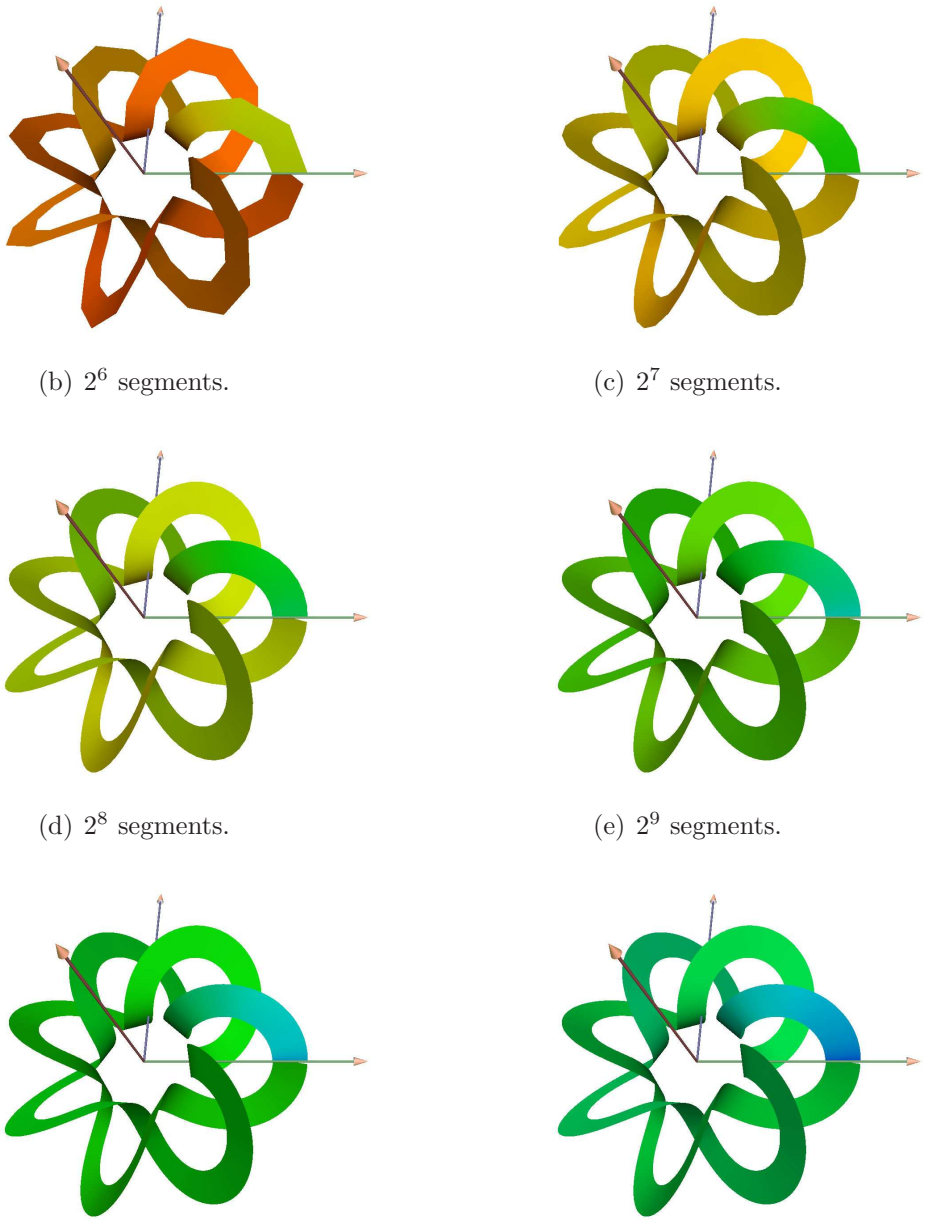
(e) 2^{10} segments.

(f) 2^{11} segments.

Figure 6.5: Color coded sweep surfaces showing the errors of double reflection method



(a) Error coding bar



(b) 2^6 segments.

(c) 2^7 segments.

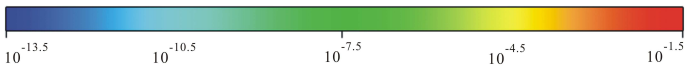
(d) 2^8 segments.

(e) 2^9 segments.

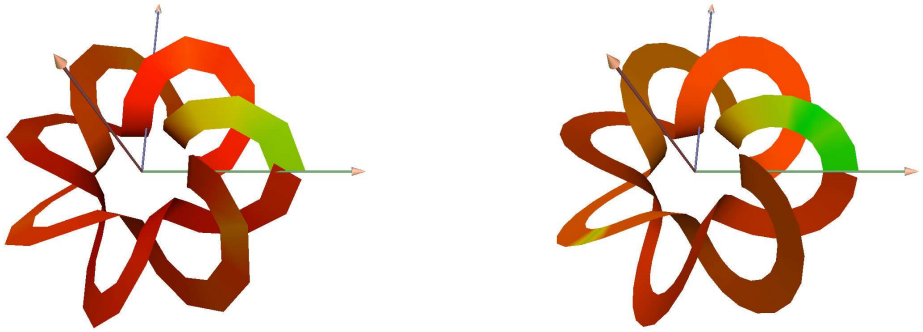
(f) 2^{10} segments.

(g) 2^{11} segments.

Figure 6.6: Color coded sweep surfaces showing the errors of 4-th order Runge-Kutta method

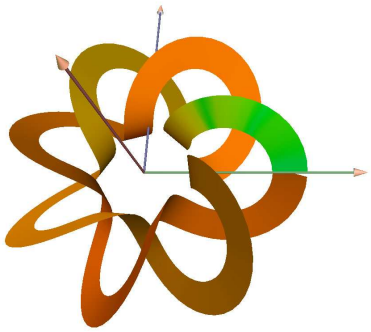


(a) Error coding bar

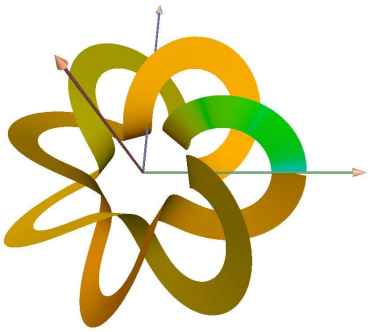


(b) 2^6 segments.

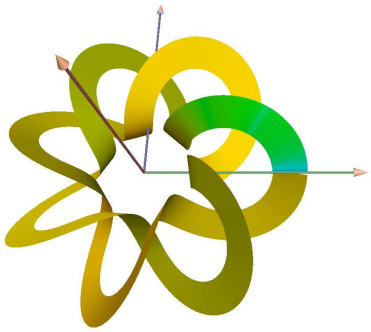
(c) 2^7 segments.



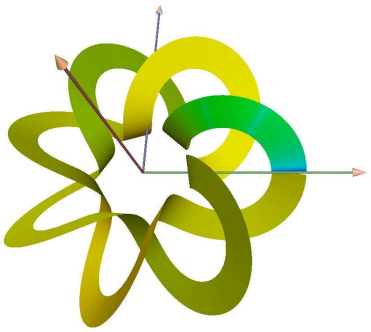
(d) 2^8 segments.



(e) 2^9 segments.

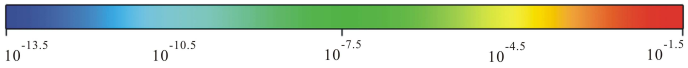


(f) 2^{10} segments.

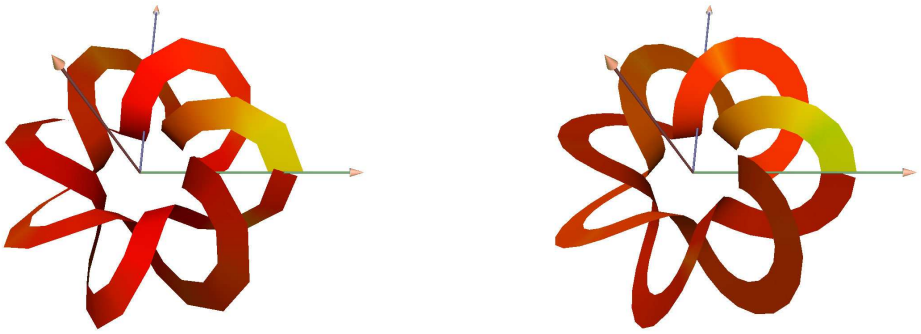


(g) 2^{11} segments.

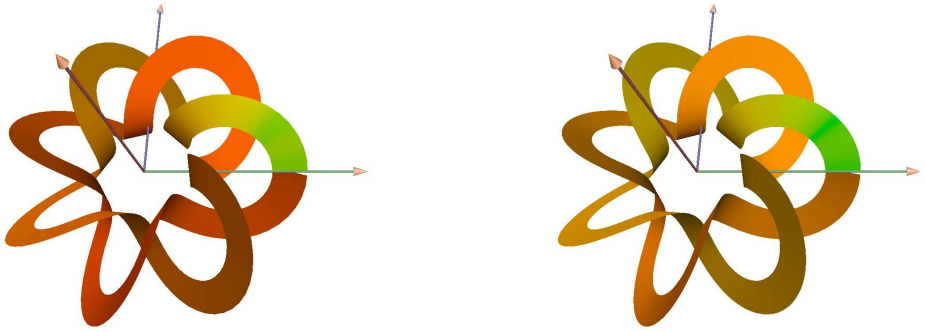
Figure 6.7: Color coded sweep surfaces showing the errors of the projection method.



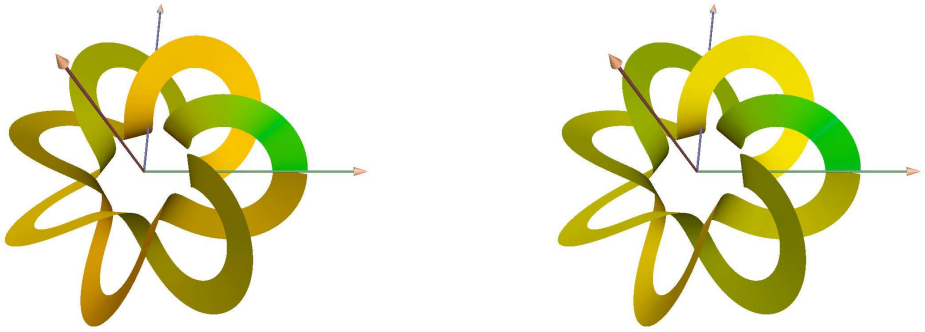
(a) Error coding bar



(b) 2^6 segments. (c) 2^7 segments.

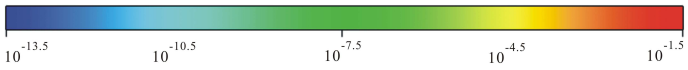


(d) 2^8 segments. (e) 2^9 segments.

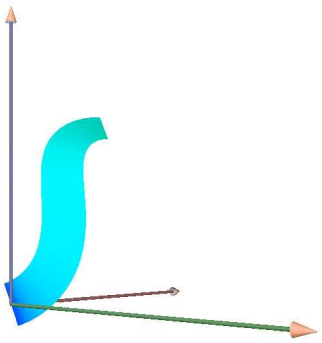


(f) 2^{10} segments. (g) 2^{11} segments.

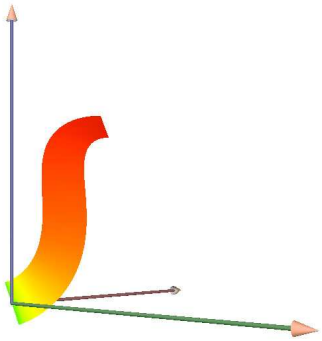
Figure 6.8: Color coded sweep surfaces showing the errors of the rotation method.



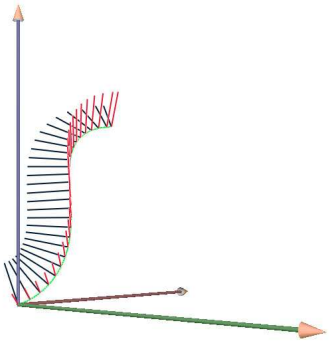
(a) Error coding bar



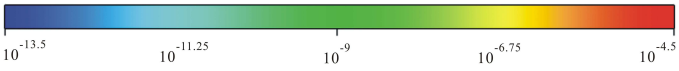
(b) Double Reflection method



(c) Rotation method



(d) Frame of Double Reflection



(e) Error coding bar

Figure 6.9: The color coded sweep surfaces showing the errors of the double reflection method and the rotation method for the PH curve in Example 2, with 256 segments.

Chapter 7

Extensions

7.1 Spine curve defined by a sequence of points

In some applications a spine curve is specified by a sequence of points $\mathbf{x}_i$ in 3D, which we may assume to lie on some unknown regularly parameterized spine curve, and we need to compute a sequence of frames U_i which has minimal rotation about the spine curve. In order to apply the double reflection method in this case, we need to furnish each data point $\mathbf{x}_i$ with a unit tangent vector $\mathbf{t}_i$.

One possible way is to define $\mathbf{t}_i$ to be unit tangent vector at $\mathbf{x}_i$ to the circle passing through the three consecutive points $\mathbf{x}_{i-1}$, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. Denote $\mathbf{a} = \mathbf{x}_i - \mathbf{x}_{i-1}$ and $\mathbf{b} = \mathbf{x}_{i+1} - \mathbf{x}_i$. It is straightforward to show that $\mathbf{t}_i = \mathbf{w}/\|\mathbf{w}\|$, where $\mathbf{w} = \|\mathbf{b}\|\mathbf{a} + \|\mathbf{a}\|\mathbf{b}$. The tangent vectors $\mathbf{t}_i$ thus defined possess the *co-sphere* property that $\mathbf{t}_i$ and $\mathbf{t}_{i+1}$ at the points $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are tangential to the sphere S determined by the four consecutive points $\mathbf{x}_{i-1}$, $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ and $\mathbf{x}_{i+2}$. This co-sphere property appears desirable because, by Theorem 5.2.3, the high accuracy of the double reflection method comes from the very same idea of computing the exact RMF of a spherical curve on a sphere determined by the data $(\mathbf{x}_i, \mathbf{t}_i; \mathbf{x}_{i+1}, \mathbf{t}_{i+1})$.

However, assuming that the $\mathbf{x}_i$ are sampled from an underlying regular curve $\mathbf{x}(u)$ with step size h, it can be shown that the global approximation error of this

scheme has the order $\mathcal{O}(h^2)$; this loss of accuracy is due that the above estimated tangent vectors t_i are only $\mathcal{O}(h^2)$ approximation to the exact unit tangent of $\mathbf{x}(u)$ at $\mathbf{x}_i$. So, an alternative is to first obtain better estimates of the unit tangent vectors t_i by using more sample points around x_i.

7.2 Using only tangent vectors

According to its defining equation (4.1), the RMF of a spine curve $\mathbf{x}(u)$ is entirely determined by the unit tangent vector $\mathbf{t}(u)$. Thus it is natural to consider computing the RMF of $\mathbf{x}(u)$ using only the sampled tangent vector $\mathbf{t}_i = \dot{\mathbf{x}}(u_i)$. From a practical point of view, this treatment is also desirable when the points $\mathbf{x}(u_i)$ are overly densely sampled, which may make the first reflection vector $\mathbf{v}_1 = \mathbf{x}_{i+1} - \mathbf{x}_i$ too small and therefore computation of the reflection R_1 unstable.

In order to apply the double reflection method in this case, all we need to do is provide a reflection vector for the first reflection R_1. Our analysis shows that the global approximation order $\mathcal{O}(h^4)$ to the true RMF of $\mathbf{x}(u)$ is preserved if the first reflection vector is chosen to be

$$\mathbf{v}_1 = 13(\mathbf{t}_i + \mathbf{t}_{i+1}) - (\mathbf{t}_{i-1} + \mathbf{t}_{i+2}). \tag{7.1}$$

Then the remaining steps of the double reflection method are the same. This assertion can be proved in a similar way to that of proving Theorem 5.6. Note that the computation of $\mathbf{v}_1$ in Eqn. (7.1) does not involve subtraction between two close quantities, and therefore is numerically robust. Note, however, a different treatment is needed to compute $\mathbf{v}_0$ and $\mathbf{v}_{n-1}$, such that an order $\mathcal{O}(h^4)$ approximations to $\mathbf{x}_1 - \mathbf{x}_0$ and $\mathbf{x}_n - \mathbf{x}_{n-1}$ are achieved. We skip the details here.

7.3 Variational principles for RMF with boundary conditions

In general, the RMF of a closed smooth spine curve does not form a closed moving frame. Therefore, when a closed moving frame with least rotation is needed, it can be generated by adding a gradual rotation to the RMF along the closed spine curve to make the resulting moving frame closed. Even for an open spine curve, it is often required that its moving frame meet given end conditions while having a natural distribution of rotation along the spine curve. So an appropriate additional rotation to the RMF needs to be computed in this case. We study in this section how this additional rotation can properly be determined.

More specifically, consider a curve segment $\mathbf{x}(s)$, $s \in [0, L]$, in arc-length parametrization. We would like to find a one-parameter family of unit vectors $\mathbf{g}(s)$ orthogonal to the tangent vector $\mathbf{t}(s)$ and satisfying the boundary conditions

$$\mathbf{g}(0) = \mathbf{g}_0 \text{ and } \mathbf{g}(L) = \mathbf{g}_1 \tag{7.2}$$

The vector $\mathbf{g}(s)$ defines an orthonormal frame $M = (\mathbf{t}, \mathbf{g}, \mathbf{t} \times \mathbf{g})$ along the spine curve.

We compare the frame M with the RMF generated by a vector $\mathbf{r}(s)$ satisfying $\mathbf{r}(0) = \mathbf{g}(0)$. Let $\alpha(s) = \angle(\mathbf{f}(s), \mathbf{g}(s))$ be the angle between the two frames, where the sign is chosen such that it corresponds to a rotation around the oriented line determined by the tangent vector $\mathbf{t}(s)$. In addition, assume that $\alpha(s)$ is continuous and satisfies $\alpha(0) = 0$. We will call $M(s)$ the *modified frame*, since it is obtained by adding a rotation of angle $\alpha(s)$ to the RMF. In this sense the RMF serves as a reference frame with respect to which another moving frame is specified.

The boundary conditions (7.2) imply that

$$\alpha(0) = 0 \quad \text{and} \quad \alpha(L) = \angle(\mathbf{f}(L), \mathbf{g}_1) + 2k\pi \tag{7.3}$$

for a some fixed integer k. The angular velocity vector of the modified frame $M(s)$ is

$$\omega_{\text{modified}}(s) = \kappa(s)\mathbf{b}(s) + \alpha'(s)\mathbf{t}(s). \tag{7.4}$$

The function $s \mapsto \alpha'(s)$ specifies the angular speed of the rotation of $M(s)$ around the tangent of the curve $\mathbf{x}(u)$. We now consider two possible ways of choosing $\alpha(s)$.

Minimum total angular speed One may choose $\alpha(s)$ that minimizes the functional

$$\int_0^L ||\omega_{\text{modified}}|| \, \mathrm{d}s = \int_0^L \sqrt{\kappa(s)^2 + \alpha'(s)^2} \, \mathrm{d}s \quad \rightarrow \quad \text{Min} \tag{7.5}$$

and satisfies the boundary conditions (7.3). Let $F(s, \alpha, \alpha') = \sqrt{\kappa^2 + \alpha'^2}$. Then we have at hand a functional of the angular function $\alpha(s)$. The moving frame $M(s)$ corresponds to a curve on the unit quaternion sphere, and minimizing the functional in (7.5) amounts to minimizing the length of this curve subject to that $\mathbf{g}(s)$ is perpendicular to $\mathbf{t}(s)$; this is the computational approach taken in [21].

Here we will analyze this variational problem to give it a simple geometric interpretation as well as an easy computational method. Solving Euler's equation of the functional (7.5) using calculus of variations yields

$$0 = F_\alpha - \frac{\mathrm{d}}{\mathrm{d}s} F_{\alpha'} = -\frac{\kappa}{(\kappa^2 + \alpha'^2)^{3/2}} (\kappa\alpha'' - \alpha'\kappa') = -\frac{\kappa^3}{(\kappa^2 + \alpha'^2)^{3/2}} \left(\frac{\alpha'}{\kappa}\right)', \tag{7.6}$$

assuming $\kappa \neq 0$. It follows that

$$\alpha'(s) = C\kappa(s) \tag{7.7}$$

for some constant C, which can be determined from the boundary conditions and the total curvature. *Consequently, the angular speed of the additional rotation around the tangent is proportional to the curvature of the curve.* Hence, minimizing

(7.5) makes the additional rotation more concentrated on curve segments of higher curvatures.

The above analysis is only valid for curved segments with $\kappa(s) \not\equiv 0$. For straight line segments, the variational problem (7.5) does not have a unique solution. In fact, the integrand in this case simplifies to $|\alpha'|$, and any monotonic function $\alpha(s)$ which satisfies the boundary conditions is a solution. Because of this non-uniqueness of solution, optimization methods as used in [21] for minimizing (7.5) will experience numerical problems with a spine curve that is close to a straight line. Based on our analysis, a more efficient method is to compute the curvatures at sampled points of the spine curve, and then distribute the additional rotation proportional to the curvatures along the curve, with respect to the RMF.

Minimum total squared angular speed One may also choose $\alpha(s)$ that minimizes

$$\int_0^L ||\omega_{\text{modified}}||^2 \mathrm{d}s = \int_0^L (\kappa(s)^2 + \alpha'(s)^2)\mathrm{d}s \quad \rightarrow \quad \text{Min} \tag{7.8}$$

and satisfies the boundary conditions (7.3). Now, with $F = \kappa^2 + \alpha'^2$, Euler's equation gives $\alpha'' = 0$, or $\alpha(s) = as$ for some constant a; that is, *the rotation of M is linearly proportional to the arc length parameter s.*

This choice of the additional rotation is not only easy to implement, but also free of the numerical problem with the method based on minimizing (7.5); so it is recommended over the first one based on minimizing the total angular speed. Note that this means of applying the additional rotation as proportional to arc-length has been suggested in the literature (e.g. [9, 53]), but here we have provided a theoretical justification from the viewpoint of the variational principle through minimization of the total squared angular speed.

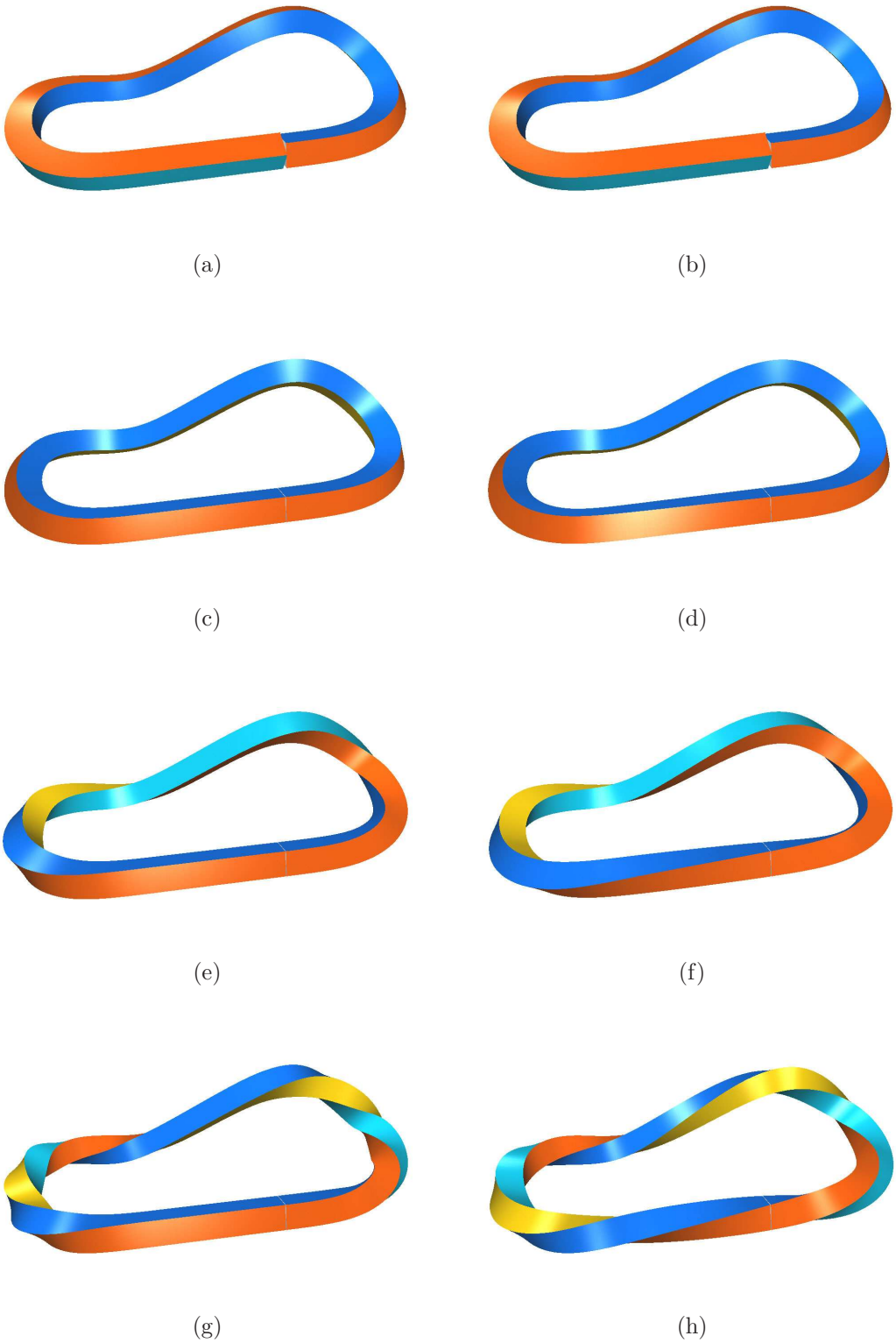
Efficient implementation of the above methods of computing a moving frame

with boundary conditions is based on angle adjustment to the RMF, either according to curvature or arclength. When the RMF is computed approximately, the resulting solution is only an approximate one. In this regard, the higher accuracy of the double reflection method makes this solution more accurate than using the projection method or the rotation method.

One may choose the integer k in (7.3) to minimize the rotation if the least deviation to the RMF is desired, or choose k to add a moving frame with a specified amount of total twist along the spine curve. Figure 7.1 shows comparison of the two methods above for computing frames meeting certain boundary conditions. The method based on total angular speed minimization (i.e., rotation proportional to curvature) and the method based in total squared angular speed minimization (i.e., rotation proportional to arclength) are shown in the first row and the second row, respectively. In each row, the four figures are for the case of using RMF computed by the double reflection method with no twist adjustment, the case of using the minimal twist to close the frame, the case of a twist of 2π, and the case of a twist of 4π. We see that the twist is more concentrated in high curvature parts of the spine curve in the first row, while it is distributed more uniformly along the curve in the second row.

A closed moving frame is useful in visualization of closed space curve, such as knots. Figures 7.2 and 7.3 show two such examples, where the closed frame is computed by adjusting an RMF by an additional rotation linearly proportional to the arclength. The curve in Figure 7.2, a cinquefoil knot, is given by

$$\mathbf{x}(t) = [\cos(t)(2-\cos(2t/(2a^2+1))), \sin(t)(2-\cos(2t/(2a^2+1))), -\sin(2t/(2a^2+1))] \tag{7.9}$$



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 7.1: Comparison in computing a closed moving frame. Minimization of total angular speed is shown in column one. Minimization of total *squared* minimization is shown in column two. In each column, from up to down, the four figures are for the case of RMF computed by the double reflection method, the case of using the minimal twist to close the frame, the case of an additional twist of 2π, and the case of an additional twist of 4π.

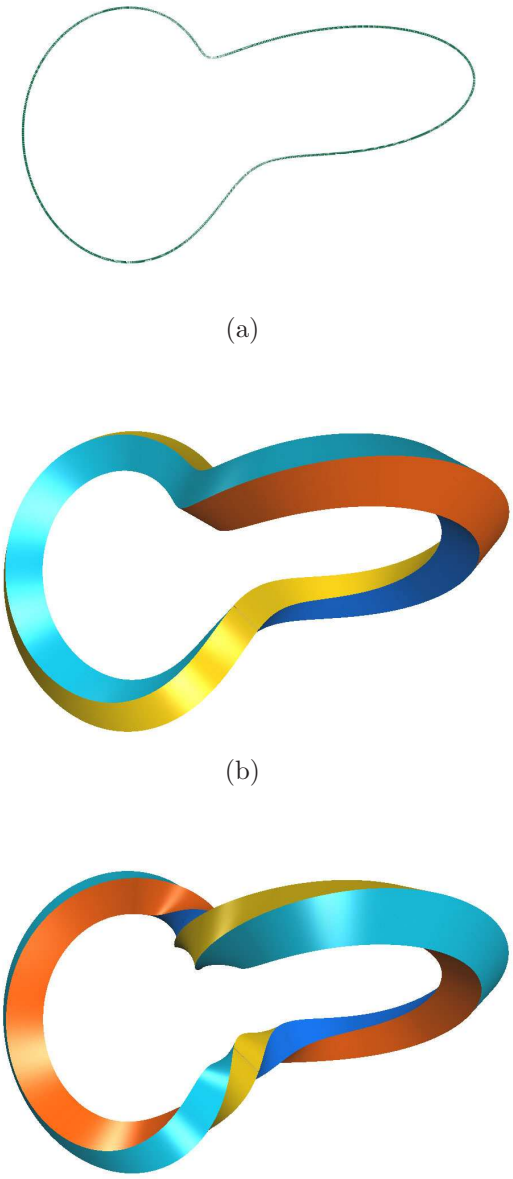
The curve in Figures 7.3, a trefoil knot, is given by

$$\begin{aligned}\mathbf{x}(t) = \quad & [41\cos(t) - 18\sin(t) - 83\cos(2t) - 83\sin(2t) - 11\cos(3t) + 27\sin(3t), \\ & 36\cos(t) + 27\sin(t) - 113\cos(2t) + 30\sin(2t) + 11\cos(3t) - 27\sin(3t), \\ & 45\sin(t) - 30\cos(2t) + 113\sin(2t) - 11\cos(3t) + 27\sin(3t)] \end{aligned} \tag{7.10}$$

Note that in these two examples the Frenet frame exhibits noticeable rotation about the curves to be visualized.

We next give two more examples of RMF based moving frame design with boundary conditions in shape modeling. In Figure 7.4, the main body of a dragon along a spine curve is modeled with a moving frame meeting user specified boundary conditions. The frame is computed using arclength twist adjustment of the approximate RMF computed by the double reflection method.

In Figure 7.5, the support structure of a glass table, as a closed sweep surface, is modeled with a moving frame meeting six conditions to make the surface have proper contact (i.e., along a line segment) with the table at three locations and with the ground at three locations. These conditions are met by adjusting an RMF by a twist linearly proportional to arclength between every two consecutive contact locations.



(a)

(b)

(c)

Figure 7.2: (a) A cinquefoil knot, given by Eqn. (7.9), is shown without visualization cue; (b) Visualization of the curve $\mathbf{x}(t)$ is enhanced by a closed sweep surface modeled using an adjusted RMF; (c) another sweep surface modeled using the Frenet frame.

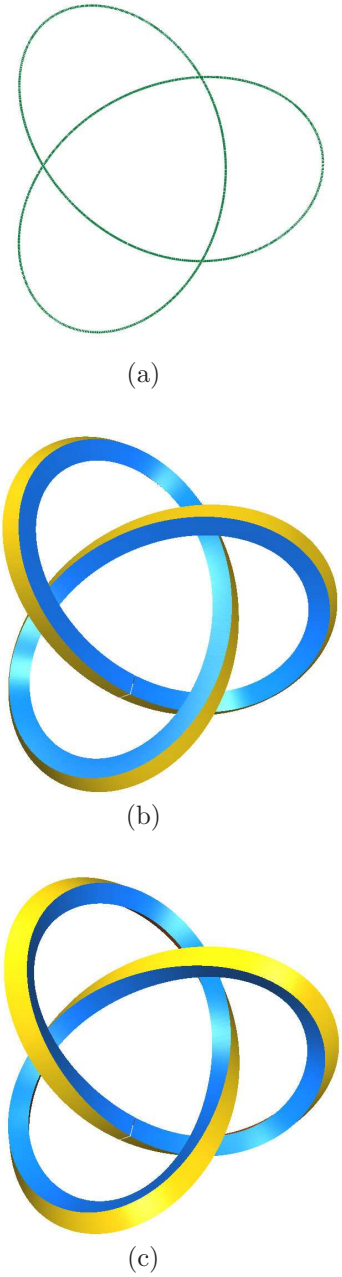
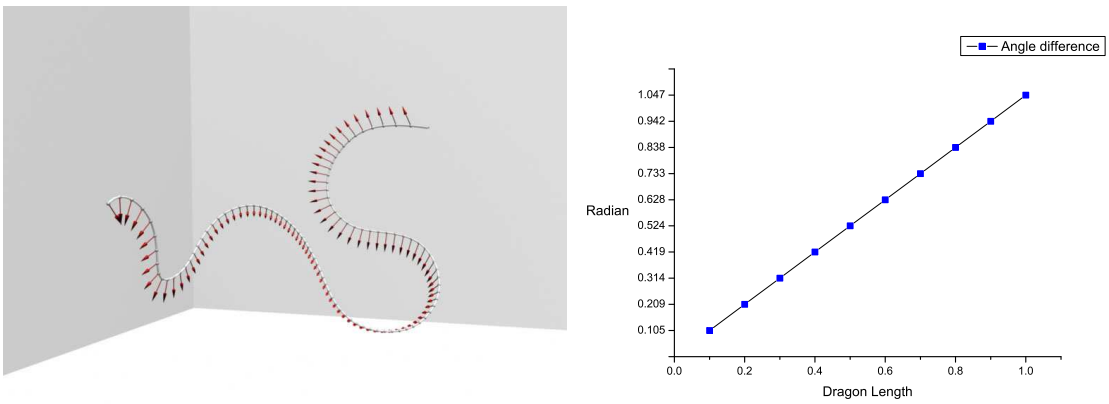


Figure 7.3: (a) A trefoil knot, given by Eqn. (7.10), is shown without visualization cue; (b) Visualization of the curve $\mathbf{x}(t)$ is enhanced by a closed sweep surface modeled using an adjusted RMF; (c) another sweep surface modeled using the Frenet frame.

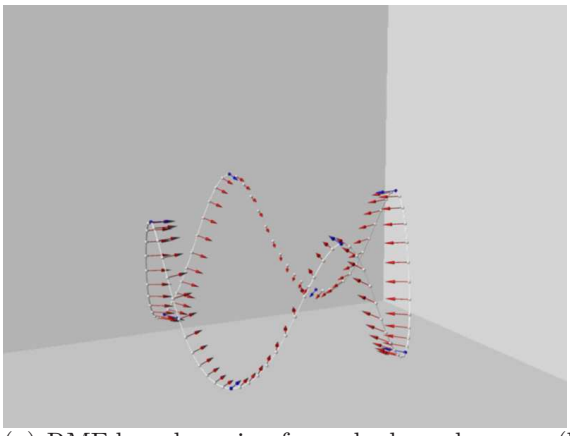


(a) RMF based moving frame by boundary condition.

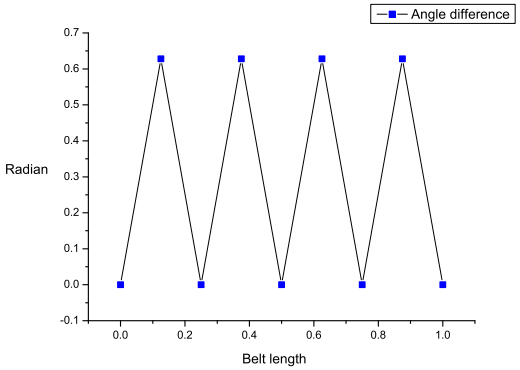
(b) Angle difference between the RMF and the frame in (a).

(c) A dragon modeled with the moving frame in (a).

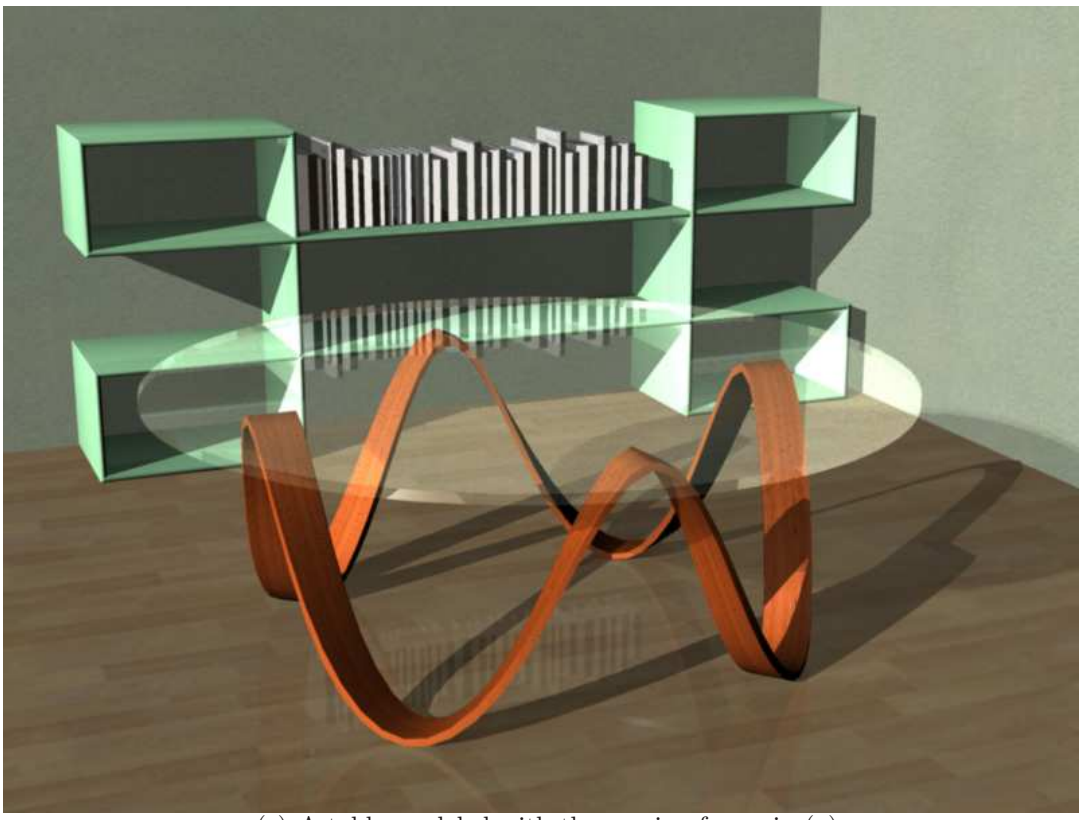
Figure 7.4: Modeling of an oriental dragon.



(a) RMF based moving frame by boundary condition.



(b) Angle difference between the RMF and the frame in (a).



(c) A table modeled with the moving frame in (a).

Figure 7.5: An RMF based moving frame is used to design the supporting structure of a glass table as a sweep surfaces

Chapter 8

Concluding remarks

We have presented a new discrete approximation method for computing the rotation minimizing frame of a space curve. The method uses two reflections in a plane to compute the next frame from the current frame, and is therefore called the *double reflection method*. This method is simple, fast, and more accurate than the projection method and the rotation method, that are currently often used in practice. We have shown that the approximation error of the double reflection method is $\mathcal{O}(h^4)$, while the errors of the other two methods are $\mathcal{O}(h^2)$, where h is the step size used to sample points on a spine curve of fixed length.

The double reflection method is also much superior to direct application of the standard 4-th order Runge-Kutta method. Although the two methods have the same order of approximation error, the double reflection method is simpler and faster, and requires only C^1 information of a spine curve, while the Runge-Kutta method needs C^2 information. We have also discussed the applications of RMF in modeling moving frames meeting boundary conditions.

Based on related research works we reviewed, we conjecture that $\mathcal{O}(h^4)$ is the maximum accuracy that can be achieved in RMF computation so far, using only the sampled positional and tangent data $(\mathbf{x}_0, \mathbf{t}_0; \mathbf{x}_1, \mathbf{t}_1)$ of a curve segment.

Chapter 9

Introduction II: vector field deformation

Shape deformation is one of the most important research fields in computer graphics. Development of this popular aspect of geometry processing diffusely affects computational geometry, simulation, CAD as well as animation. Based on the common requests of users, there are mainly two core problems involved in this key research field: one is non-self-intersecting, the other is volume preserving. Smoothness-preserving and easy user-interaction are also generally required. To satisfy these constraints, several advanced algorithms, like the well-known free-form deformation(FFD) [43], manifold extended method, Laplacian deformation [48] as well as vector field deformation are developed.

Vector field deformation is not only applied to shape deformation but also to texture synthesis, non-photorealistic rendering, and fluid simulation [54]. In [17], Davis introduces an important property of divergence-free vector field: *deformation under divergence-free vector field is volume-preserving.* [50] mentions that no self-intersection occurs in divergence-free vector field since path lines of field have no intersection with each other in the domain. [52] shows that divergence-free vector field deformation is C^1 continuous, which means smoothness-preserving. Such

deformation also preserves detailed features. It is obvious that the properties of divergence-free vector field are quite suitable to be used in deformation.

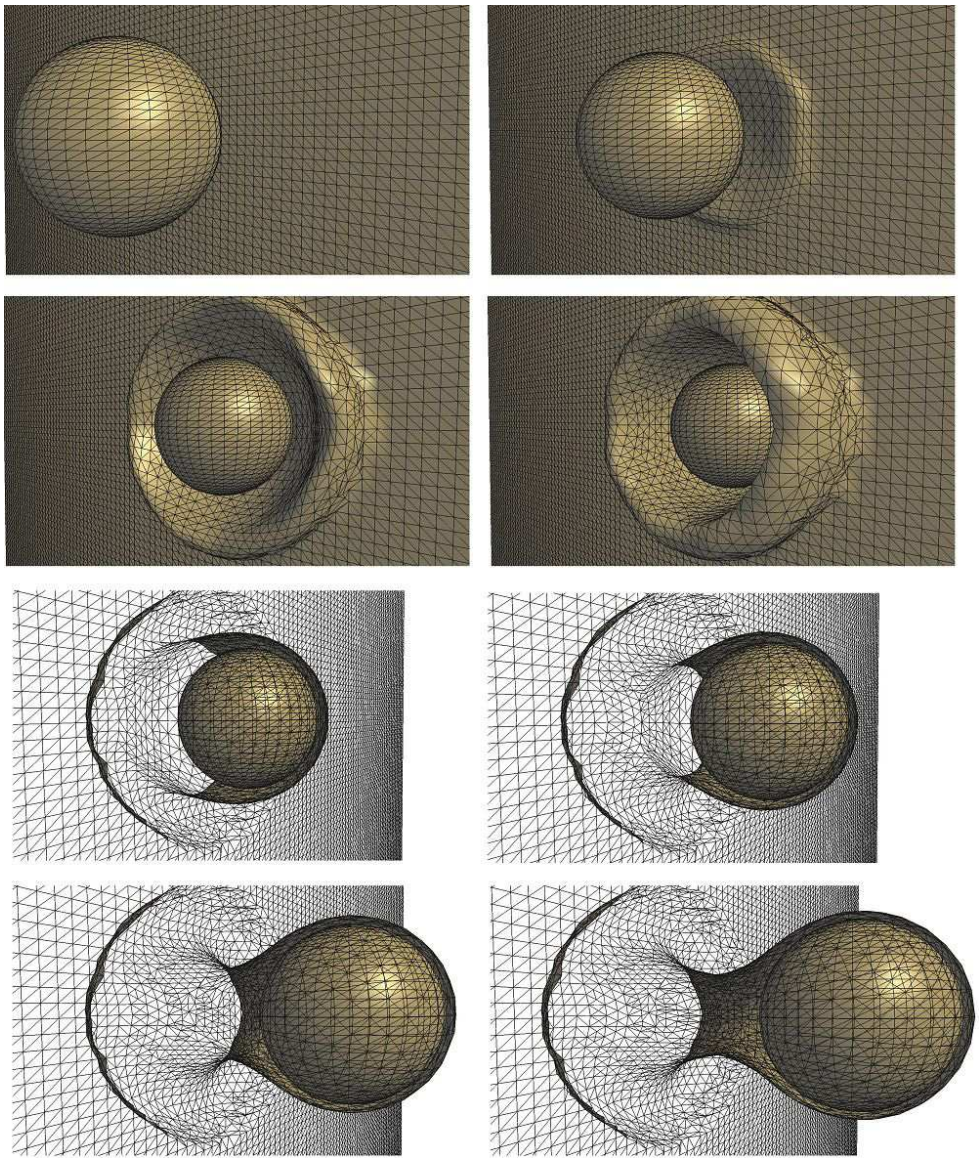


Figure 9.1: Curl vector field deformation example: when the sphere keeps moving, curl vector field around the sphere occurs volume-preserving deformation on the surface in realtime, without GPU speed-up technology.

[52] also provides an innovative algorithm to construct divergence-free vector field for volume-preserving deformation. The basic idea is that a divergence-free vector field can be constructed from the cross product of the gradients of two scalar fields:

$v(x, y, z) = \nabla p(x, y, z) \times \nabla q(x, y, z)$. It provides spherical vector field primitive for dragging and cylindrical vector field primitive for bending.

The design of vector field primitives in [52] is, however, based on pre-defined shapes but not related to object geometry. This kind of design may work well on objects which are spherical or cylindrical but not on free-form objects. Our contribution is to introduce a new method of divergence-free vector field primitive construction for general shapes. The spherical vector field primitive in [52] becomes a special case in our method. The new method is based on distance field rather than pre-defined shapes so that object's geometric information will be used in the vector field design. Figure 9.1 shows a vector field deformation example in our method.

Chapter 10

Related work II

Shape deformation, especially 3D deformation is a hot research field in computer graphics. Early surface deformation approaches can be classified by their control handles. [43] introduces the famous lattice free-form deformation(FFD) technique. It is extended by [16] by using non-parallelepipedical lattices. Lattice handle is good at global shape control but has problems with local detail adjustment. New algorithm based on wire handle or vertex handle are developed to deal with this problem. [4] and [47] exploited new deformable proxy constructed by wire handle with domain curves in shape deformation. For accurate manipulation, [25] and [24] define multi-level boundary representations for modeling primitives and develop vertex-based controlling algorithm from lattice-based FFD algorithm.

Generally speaking, classical FFD method and their variants usually require complex user-interaction [11]. More advanced deformation method are developed to simplify controlling process and to satisfy given constraints by designing different basis functions. [12] uses triharmonic radial basis functions for all kinds of scattered data interpolation problems while [2] defines a new shape deformation tool named swirling-sweeper on their framework. It is based on blending of several rotations whose magnitude decreases away from control center.

Laplacian approach is well developed in shape deformation since Laplacian coordinate represents surface details by local mean [1] [36]. [48] discuss surface editing over an intrinsic surface representation of a surface. [55] extends the idea to volumetric domain to solve the problem of large deformations. Vector field is widely used in both 2D/3D shape deformation, as well as fluid simulation [49], visualization [51] and texture synthesis [42]. [54] presents a vector field design system to create a wide variety of vector fields that allow user to control over the vector field topology. [52] defines a new shape deformation method based on time-dependent divergence-free vector fields. We call it cross-product construction. The deformation holds prosperities such as self-intersection-free, volume-preserving, smoothness preserving and feature preserving.

We notice that the cross-product construction only create spherical vector field. it presents a multiply spherical vector field combination for free-form objects which has limitations. In our research, we first extend the idea from spherical vector field to strip vector field construction, then we present a new vector field construction named curl field construction for free-form shape objects which will be more flexible in applications.

Chapter 11

Cross-product vector field deformation

11.1 Introduction of cross-product vector field construction

cross-product vector field deformation was first presented by Wolfram Von Funck in 2006 [52]. We have extend the idea more widely in our research. Before we discuss the extension of cross-product vector field construction, we would like to introduce the original cross-product vector field construction first, which is the core of the approach.

Let p_x and p_y denote the partial derivatives $\frac{\partial p}{\partial x}$ and $\frac{\partial p}{\partial x}$. Then the divergence-free vector field v can be constructed from the gradients of two scalar fields $p(x, y, z)$ and $q(x, y, z)$ as

$$v(x, y, z) = \nabla p(x, y, z) \times \nabla q(x, y, z)$$

In the cross-product vector field construction, there is three kinds of region: **inner region**,**outer region** and **intermediate region**. Inner region is a constant or linear field, describing a simple translation; outer region has a zero deformation; intermediate region is the blending region between inner region and outer region.

The different regions are specified by defining a scalar region field $r(\mathbb{X})$ with $\mathbb{X} = (x, y, z)$ and two thresholds $r_i < r_o$. If $r(\mathbb{X}) < r_i$ then point $\mathbb{X}$ is in inner region; if $r(\mathbb{X}) > r_o$ then point $\mathbb{X}$ is in outer region; if $r(\mathbb{X}) < r(\mathbb{X}) < r_i$ then point $\mathbb{X}$ is in inner region.

Let e(x) and f(x) denote two C^2 continuous scalar fields. With the definitions above we can define the scalar fields p and q as

$$p(\mathbb{X}) = \begin{cases} e(\mathbb{X}), & if \quad r(\mathbb{X}) < r_i \\ (1-b)\cdot e(\mathbb{X}) + b \cdot 0, & if \quad r_i \le r(\mathbb{X}) < r_0 \\ 0, & if \quad r_0 \le r(\mathbb{X}) \end{cases}$$

$$q(\mathbb{X}) = \begin{cases} f(\mathbb{X}), & if \quad r(\mathbb{X}) < r_i \\ (1-b)\cdot f(\mathbb{X}) + b \cdot 0, & if \quad r_i \le r(\mathbb{X}) < r_0 \\ 0, & if \quad r_0 \le r(\mathbb{X}) \end{cases}$$

$b = b(r(\mathbb{X}))$ is a blending function given in *Bezier* representation as follows where B_i^4 are the Bernstein polynomials, $w_0 = w_1 = w_2 = 0$ and $w_3 = w_4 = 1$:

$$b(r) = \Sigma_{i=0}^{4} w_i B_i^4(\tfrac{r-r_i}{r_o-r_i})$$

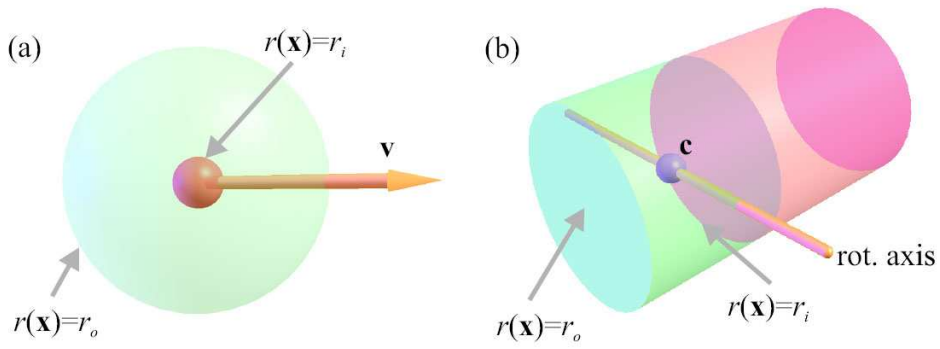


Figure 11.1: (a) is spherical translation field and (b) is cylinder rotation field inside the inner region.

For spherical translation field, the linear scalar fields $e(\mathbb{X})$ and $f(\mathbb{X})$ are defined as follows where u and w are two arbitrary orthogonal vectors

$$e(\mathbb{X}) = u(x-c)^T,\ f(\mathbb{X}) = w(x-c)^T$$

For cylinder rotaion field, the linear scalar fields $e(\mathbb{X})$ and $f(\mathbb{X})$ are defined as

follows where a are the normalized axis direction

$$e(\mathbb{X}) = a(x-c)^T,\ f(\mathbb{X}) = (a \times (x-c)^T)$$

11.2 Extension of cross-product vector field construction

Generally, vector field deformation contains two kinds of meshes: deforming mesh and deformed mesh. A deforming mesh is the base of divergence-free vector field. It provides shape information for vector field construction. Vector field keeps the same in object space while there is no change at deforming mesh but will follow the moving of deforming mesh in world space. The deformed mesh is for deforming effect performance under vector field.

In [52], elements of divergence-free vector field are defined for regular shapes such as spheres and cylinders. As observed in experiments, we notice that the structure of spherical divergence-free vector field is closely correlative to the distance from the pre-defined center which means spherical divergence vector field will have obvious different speed in surface of extremely non-spherical models. In vector field, a reasonable requirement is that vector field should have similar speed at similar altitude of moving direction on object surfaces and offset surfaces. This property will keep smooth effect in vector field deformation.

The sum of divergence-free vector fields is still a divergence-free vector field. When we construct a particular divergence-free field for an complicate object, we can first divide the object into many elements. Each element is covered with an element divergence-free vector field. After that, the constituent vector fields can be added together.

The first contribution of this part is the extension of cross-product divergence

vector field constructions for three object elements: strip, sphere and torus. There are two kinds of actions for strip type object in deformation: sweeping and sticking. when head of strip shape object points to face direction, the former action moves ahead, the latter action moves aside. We provide time-independent vector field same as that in [52].

11.2.1 Extended spherical divergence free vector field

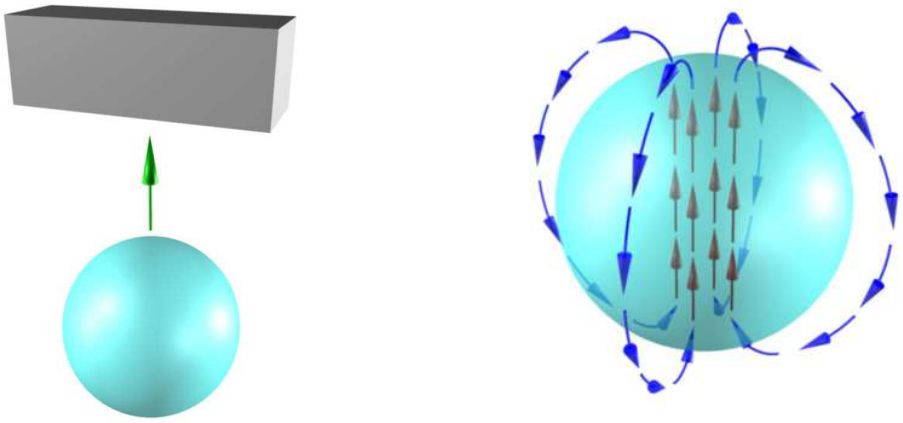


Figure 11.2: Spherical vector field illustration. Green arrow is moving direction, red arrows locate inside of object, blue arrows locate outside of object.

In the construction of divergence-free vector field, because scalar attenuation can effectively control the shape of the vector fields, we use cross-product of gradients of two scalar fields $p(x, y, z)$ and $q(x, y, z)$ to constructed a 3D divergence-free vector field, as what has been introduced in [52]:

$$V(x, y, z) = \nabla p(x, y, z) \times \nabla q(x, y, z)$$

Beginning from sphere primitives will help to understand our method. [52] has introduced a divergence-free vector field construction for spherical primitives. Our extended spherical vector field construction is similar to the vector field construction in [52]:

$$\begin{cases} v &= \nabla p \times \nabla q \\ p &= (1 - dist(\mathbb{X})) * N \cdot (\mathbb{X} - \mathbb{C}) \\ q &= (1 - dist(\mathbb{X})) * B \cdot (\mathbb{X} - \mathbb{C}) \end{cases}$$

$\mathbb{X}$ represents positions of vertices. $dist(\mathbb{X})$ returns distance value of vertices to deforming object. $\mathbb{C}$ is the center of deforming object.Normalize vector N, B and moving direction V_dir construct orthogonal coordinate system. v gives the vector value of $\mathbb{X}$ in vector field.

The difference of this spherical vector field construction compared to method in [52] is that we introduce distance field of deforming object into our method instead of using distance to center point directly. Distance field is good at representing object features so we get better result from the extended spherical vector field construction [fig].

11.2.2 Divergence free vector field for strip: sticking action

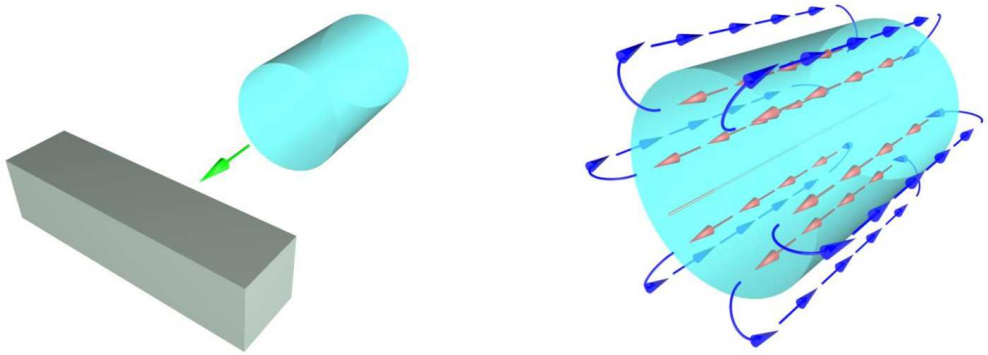


Figure 11.3: Strip type-I vector field illustration. Green arrow is moving direction, red arrows locate inside of object, blue arrows locate outside of object.

There are two kinds of definitions for strip type divergence-free vector field. Type-I defines a divergence-free field moving ahead, type-II defines a divergence-free field moving aside. For a type-I vector field, the vector flows along the strip surface

in inner region at moving direction,then flows out of strip object from the head and flow back along external surface in near-outer region. Finally it flows into the strip type object from the tail again. The outer region of vector field will be attenuated to zero. We bind the space around strip object to an inner curve, which locates at the center position of object. The curve will have retraction at the end for smooth diversification in vector field. Each vertex in the space will be attached a distance value to the closest point of curve $curve(\mathbb{X})$. Then we construct scalar field $p(x, y, z)$ and $q(x, y, z)$ based on distance field $dist(\mathbb{X})$ and rotation minimizing frame []:

$$\begin{cases} v &= \nabla p \times \nabla q \\ p &= (1 - dist(\mathbb{X})) * N \cdot (\mathbb{X} - curve(\mathbb{X})) \\ q &= (1 - dist(\mathbb{X})) * B \cdot (\mathbb{X} - curve(\mathbb{X})) \end{cases}$$

N and B are the components of rotation minimizing frame. Instead of the computing of integral definition, we implement the double-reflection method, a new four-order approximated numerical rotation minimizing frame method in order to construct the vector field at real time.

The spherical divergence-free vector field [52] is a special case of type-I divergence-free vector field. If the inner curve degenerates to a center point, the vector field will degenerate to be the spherical vector field in [52].

11.2.3 Divergence free vector field for strip: sweeping action

Now we are going to construct the type-II divergence-free vector field. In this kind of field, vector flow along the section in inner region, flows out of strip object from one side and then flows into strip object from another side along the external surface of strip. Outer region of the vector field will be attenuated to zero as

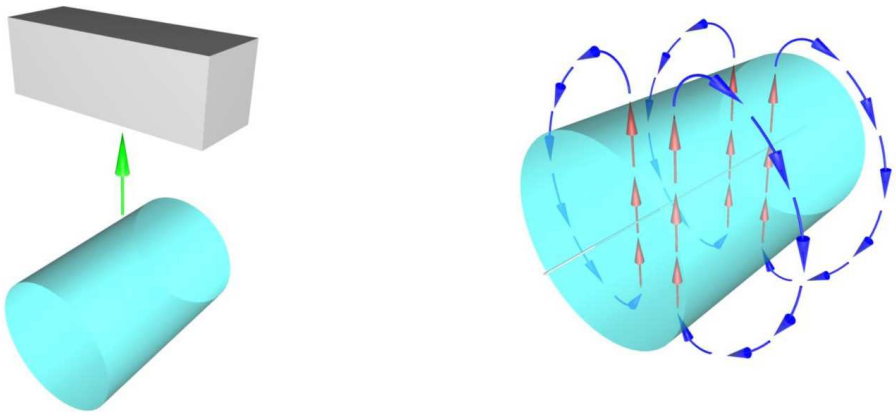


Figure 11.4: Strip type-II vector field illustration. Green arrow is moving direction, red arrows locate inside of object, blue arrows locate outside of object.

well. We also bind the space around strip type object to an inner retractive curve, which locates at the center position. Every point on the curve has been attached a scalar value $scalar(\mathbb{X})$, increasing from head to tail. Each vertex on the surface will be assigned the same scalar value related to the closest point on the curve. To construct the second scalar field, we use the distance field of the strip and the distance of each vertex to the closest point of inner curve:

$$\begin{cases} v &= \nabla p \times \nabla q \\ p &= scalar(\mathbb{X}) \\ q &= (1 - dist(\mathbb{X})) * U \cdot (\mathbb{X} - curve(\mathbb{X})) \end{cases}$$

$dist(\mathbb{X})$ is the distance field of strip object. U is a vector perpendicular to moving direction on each section face. Each object has an axis curve inside it. $curve(\mathbb{X})$ returns the closest point of inner curve. $scalar(\mathbb{X})$ returns the attached scalar value.

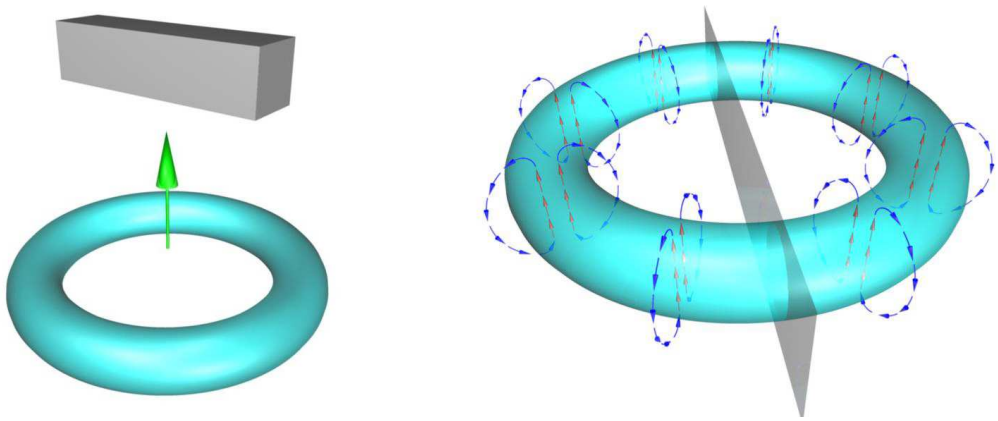


Figure 11.5: Torus vector field illustration. Green arrow is moving direction, red arrows locate inside of object, blue arrows locate outside of object.

11.2.4 Divergence free vector field for torus cases

A torus is a special case of a strip. In strip type vector field construction, one of the scalar fields is increased along inner axis curve from tail to head. However, a torus has no tail and head and there is no such continuous scalar field along inner axis curve. In our method, the torus can be divided into two pieces of strips. In order to provide a divergence-free vector field for torus, we combine two strip type vector fields. It is well known that the summation of multiply divergence-free field is also a divergence free field, so when we construct vector field for a complicated object, we can divide it into parts with regular shape so that we can cover each part with vector field construction. For a torus, it would be covered with two modified strip vector fields.

This modified strip divergence free vector field for each part is similar to type-II strip vector field. The feature is that vector flows on surface at the end of each strip, direction are parallel to the end surface so two strip vector fields will have strict and non-intersectant combination at the separate position. We also attach increasing scalar value $scalar(\mathbb{X})$ to the inner curve that any point in space will

attach the same scalar value from the closest point of curve. Because the inner curve stops at the two ends, so the scalar value which is out of end surface will stay the same, which means the gradient of the scalar field will be zero at this zone. This is the trick to keep the direction of vector flow at the end parallel to the surface but not leak out. Now we define the scalar field as the second scalar field in type-II construction: first, $curve(\mathbb{X})$ returns the closest point position of inner curve; second, $dist(\mathbb{X})$ returns the value of distance field. $\mathbb{X} - curve(\mathbb{X})$ is the vector from curve to current point $\mathbb{X}$. We multiply attenuation parameter $(1 - dist(\mathbb{X}))$ to the dot product of vector U and $(\mathbb{X} - curve(\mathbb{X}))$, then we get the second scalar field. On each section perpendicular to the inner curve, the vector flow contains two circle loops which presents an Arabia number "8"

$$\begin{cases} v &= \Sigma v_i \\ v_i &= \nabla p \times \nabla q \\ p &= scalar(\mathbb{X}) \\ q &= (1 - dist(\mathbb{X})) * U \cdot (\mathbb{X} - curve(\mathbb{X})) \end{cases}$$

11.2.5 Approximate vector field construction for free-form models

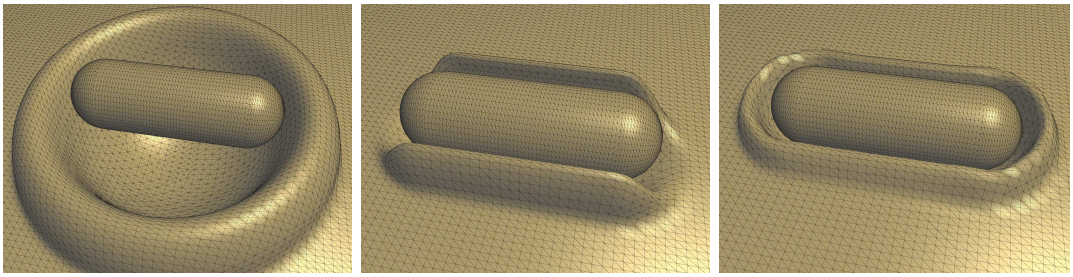


Figure 11.6: Sphere, strip and free-form vector field deformation.

Besides the three kinds of element vector field definitions, we also introduce an advanced construction for free-form objects. We notice that sphere divergence-free vector field cannot cover a free-form object tightly. Strip divergence-free vector field has a better performance at non-spherical object, but it will have less

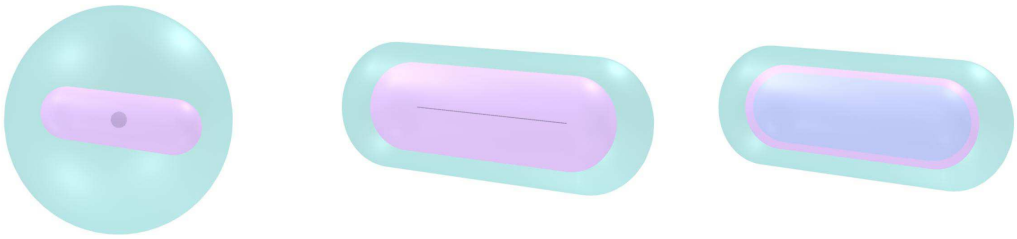


Figure 11.7: Vector field illumination. Pinky layer is the shape of objects; green layer is the range of vector fields; blue layer is the pivot area.

movement at the two ends. We expect the vector field of free-form objects has well-proportioned vector flow around to make deformation natural. To design of such a vector field for free-form objects, we should take geometry into consideration. An important feature is that the moving speed in vector field should related to the distance to object surface. This feature ensures that the flow around objects has similar movement.

To discover a general construction for free-form objects is difficult because of the complexity of object shape. However, we can consider an approximated free-form vector field construction, which get useful condition to make construction and has acceptable approximate volume-preserving feature with general movement. To construct such a field, we use an offset surface inside the object. Each point in the space has a closest position on the surface, which we name "pivot", defined as follows: U and W are orthogonal vectors, at the same time, both of them are perpendicular to the moving direction. Vector field p is provided by U with distance field attenuation. It also contains a turning feature p_{spec} to adjust the direction, which is constructed by gradient and vector to pivot. Vector field q has the similar construction.

Finally we get our free-form vector field construction from the cross product of vector fields p and q.

$$\begin{cases} v & = & p \times q \\ p & = & U * (1 - dist(\mathbb{X})) - p_{spec} \\ q & = & W * (1 - dist(\mathbb{X})) - q_{spec} \\ p_{spec} & = & gradient(\mathbb{X}) * U \cdot (\mathbb{X} - pivot(\mathbb{X})) \\ q_{spec} & = & gradient(\mathbb{X}) * W \cdot (\mathbb{X} - pivot(\mathbb{X})) \\ pivot(\mathbb{X}) & = & \mathbb{X} - gradient(\mathbb{X}) * (dist(\mathbb{X}) + d_{off}) \end{cases}$$

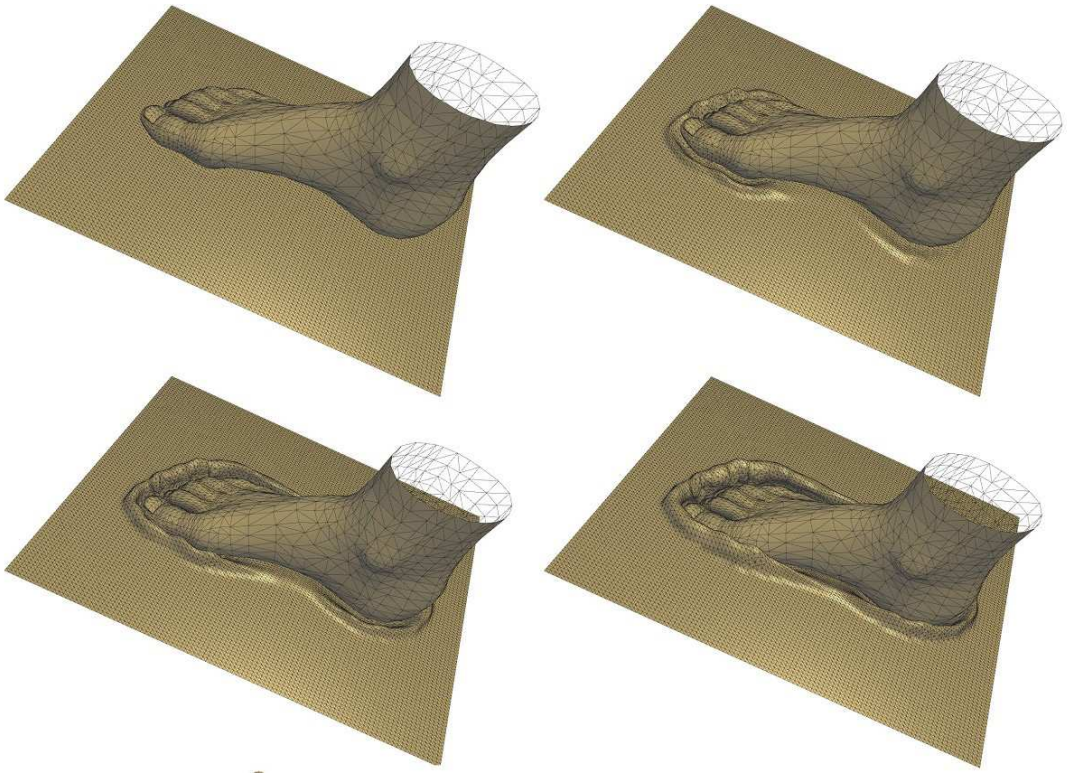


Figure 11.8: Deformation of Complicate objects: foot print on earth.

11.2.6 Advanced discussion

Stamper effect

The disadvantage of the above methods is that there is a gap between the deforming objects and the deformed objects. It is because that when vector field object gets close to the deformed object the flow of vector field would not allow their surface to touch each other. This is acceptable when the object surface is smooth, however, it will not be so good when the surface contains details. For instance, a

coin.

We modify the above method a little bit to solve this problem. This method is also volume preserving, but the colliding surfaces can touch each other. The deformation will only go into effect when collision happens. The deformed object will get two kinds of effects from the deforming object. One is denting effect, which impacts the surface of the deformed object. The surface will be depressed as far as the deforming object moves. Suppose at each frame, a vector field object will move δd and area of collided surface of deformed object is δs. Then the depressed volume is $\delta v = \delta s \times \delta d$. The other effect is protruding effect, which impacts the area around the dented surface on the deformed object. The area of protruding surface is δs_p.

Although deformation is operated by vector field, what we need to do is keep the volume dent speed equal to the volume protruding speed. That means at each frame, the dent volume should be equal to the protruding volume. We know that the sunken volume is $\delta v = \delta s \times \delta d$. For the protruding volume, it is $v_{prot} = \int v_{ds} ds$. For mesh objects, $v_{prot} \approx \sum s_{tri} \times v_{bary} \times t_{bary}$, where s_{tri} is size of each triangle, v_{bary} is the speed of barycenter in the vector field. Then we need to decide the moving time t_{bary}. As what we discussed above, $\sum s_{tri} \times v_{bary} \times t_{bary} = \delta s \times \delta d$, So $t_{bary} = \delta s \times \delta d \setminus (\sum s_{tri} \times v_{bary})$

Deformation and self-deformation

Sometimes a deforming mesh also requires deformation in animation. For example, when a hand grasps a piece of plasticine, the hand mesh also needs deformation when it deforms plasticine at the same time. Vector field deformation is useful to

meet such requirements.

In order to keep the topology feature of objects in divergence-free vector field construction, we introduce integrated vector field construction into our algorithm. We have known that the divergence vector field has good properties for deformation. However, some process of vector field construction such as distance field computing is time-consuming. In free-form vector field construction, the distance field needs to be re-computed whenever the shape of deforming model is changed, which implies heavy computation. In our intergrated vector field construction, element vector field only compute once by before deformation to avoid time consuming. We use the three kinds of element vector field construction defined above to construct intergrated vector field, as well as the spherical vector field and cylindrical vector field introduced in [52].

In this method, a complicated object will be divided into some object elements. Each object is covered with a divergence-free element vector field. Nearby element vector field overlap each other. Because element vector fields are divergence-free and continuous, the intergrated vector field is also divergence-free and continuous.

The distance field of required element objects are computed and stored before deformation. These stored distance fields can provide necessary information for vector field construction in real-time. An element vector field will remain the same in its object space in deformation. Multiplied with the inverse transformation matrix of element vector fields, vertices will be translated to relative element vector field to obtain relative deformation. It is well known that the sum of divergence vector fields are also divergence vector field. So intergrated vector field construction provides us with fast-computing volume-preserving vector fields for free-form objects.

Chapter 12

Curl vector field deformation

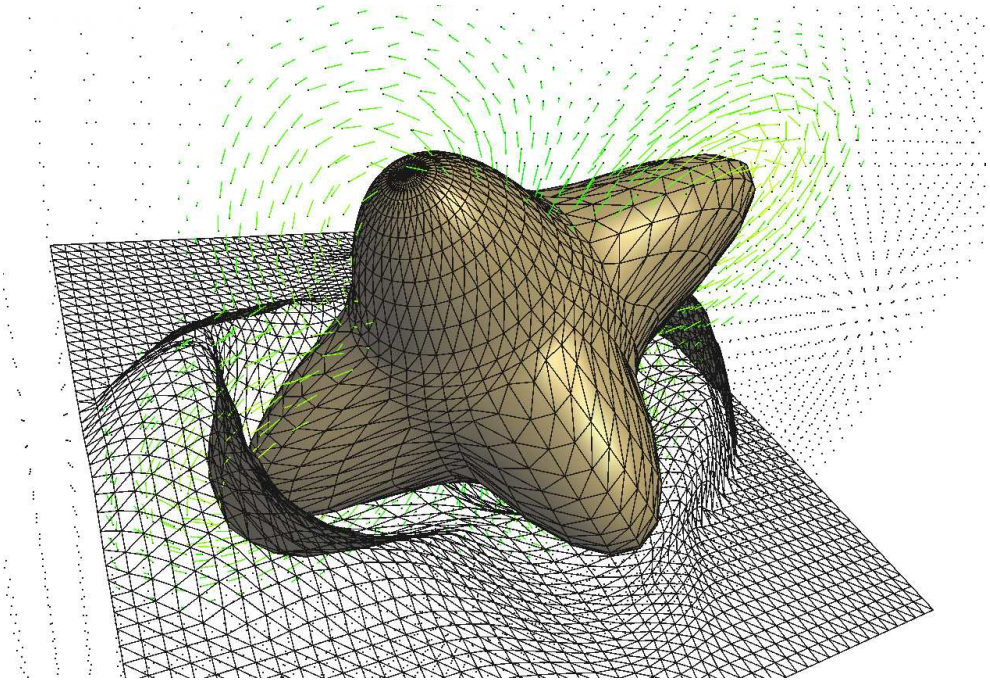


Figure 12.1: Illustration of vector flow in curl vector field. The colored vectors on the section plane show the directions and magnitude of curl vector flow. The curl vector field has the same shape feature as the shape feature of the deforming object.

The goal of the approach proposed here is to construct a special vector field effectively from the curl of a base vector filed to provide non-self-intersecting and volume preserving deformation. Generally, vector field deformation involves two

objects: the deforming object and the deformed object. The deforming object that provides shape information is the base of divergence-free vector field construction.

The deformed object is the object being deformed under the vector field of the deforming object. For some special cases, a vector fields can be generated based on pre-defined conditions without any deforming object. However, the deformed object is always required.

Mathematically, a divergence-free vector field has some special features. One of them is that objects deformed under divergence-free vector field is volume preserving, which is a very important feature in mesh deformation, and it is the basic criterion of our algorithm. From previous research, we know that divergence-free vector field can be constructed from the gradients of two scalar fields $p(x, y, z), q(x, y, z)$ in 3D or construct it as a co-gradient field of a scalar field $p(x, y)$ in 2D [52]. The construction is based on a well-known feature that cross product of the gradients of two scalar fields is divergence-free [17].

Here, we present a new way of construction which is more flexible and easily understood for 3D divergence-free vector field construction from the curl field. It is also well-known that the divergence of a curl field is equal to zero.

$$\mathbf{div}(\mathbf{curl}(v)) = \nabla \cdot (\nabla \times v) = 0$$

The above feature ensures that our vector field construction based on curl will be volume preserving.

12.1 Translation

First, we construct a base vector field for translation. A local coordinate framework $[i, j, k]$ is built around the deforming object for vector field construction.

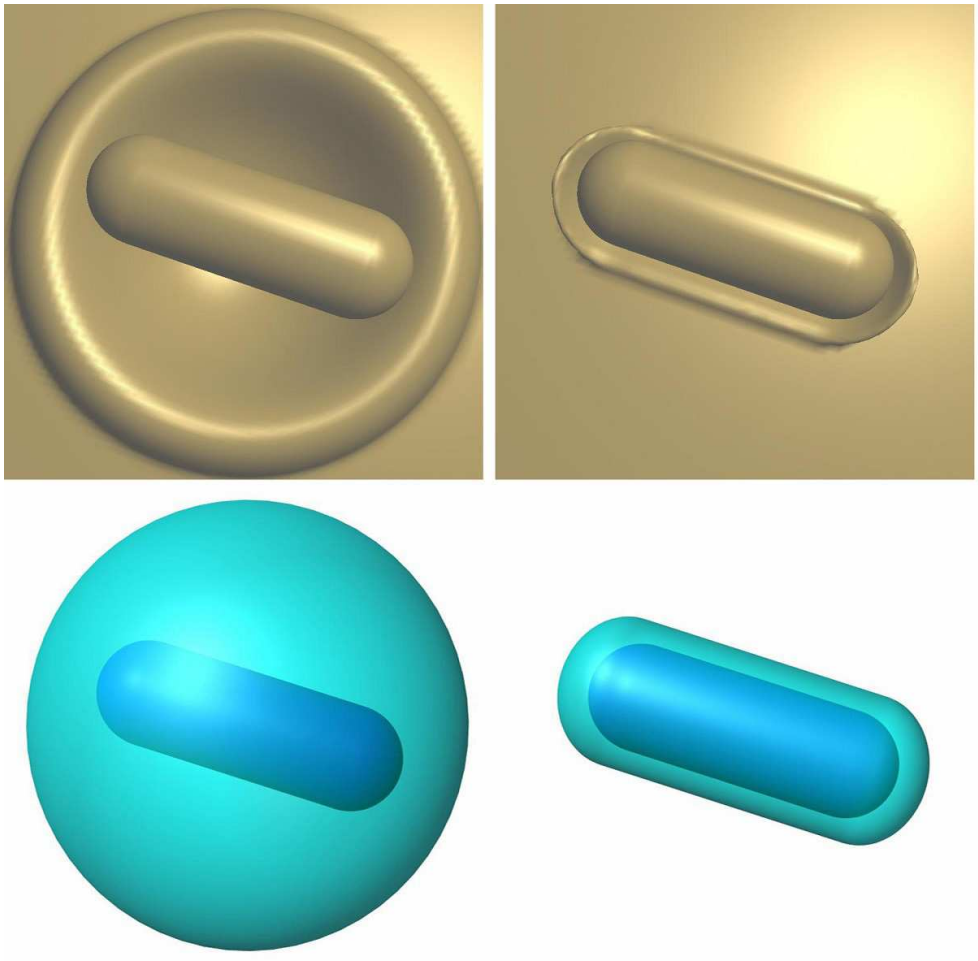


Figure 12.2: Comparison of different divergence-free vector field construction algorithm: The light blue color component illustrates the shape of the vector fields. The left column is constructed by [52], which only relates to a pre-defined field radius; the right column is constructed by the new algorithm, which relates to the distance field of deforming object.

i, j, k are three orthogonal unit vectors. i denotes the moving direction of deforming object when j and k are freely chosen. Vertex X will be denoted as X' in local coordinate system. For each deforming object, we define a base vector field $\mathbf{B_T}$ rotating around its i axis in local coordinate. Let P be (0,y,z), we have:

$$\mathbf{B_T}(X) = P \times i \tag{12.1}$$

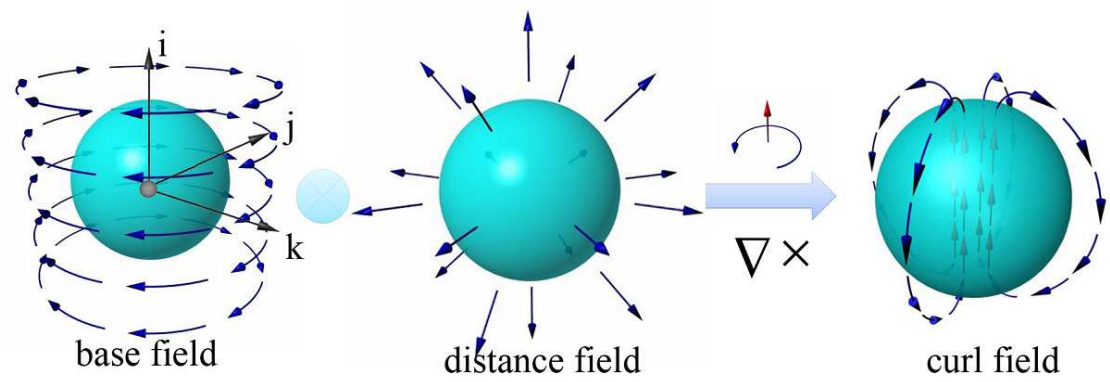


Figure 12.3: Illustration of curl vector field generation for translational motion.

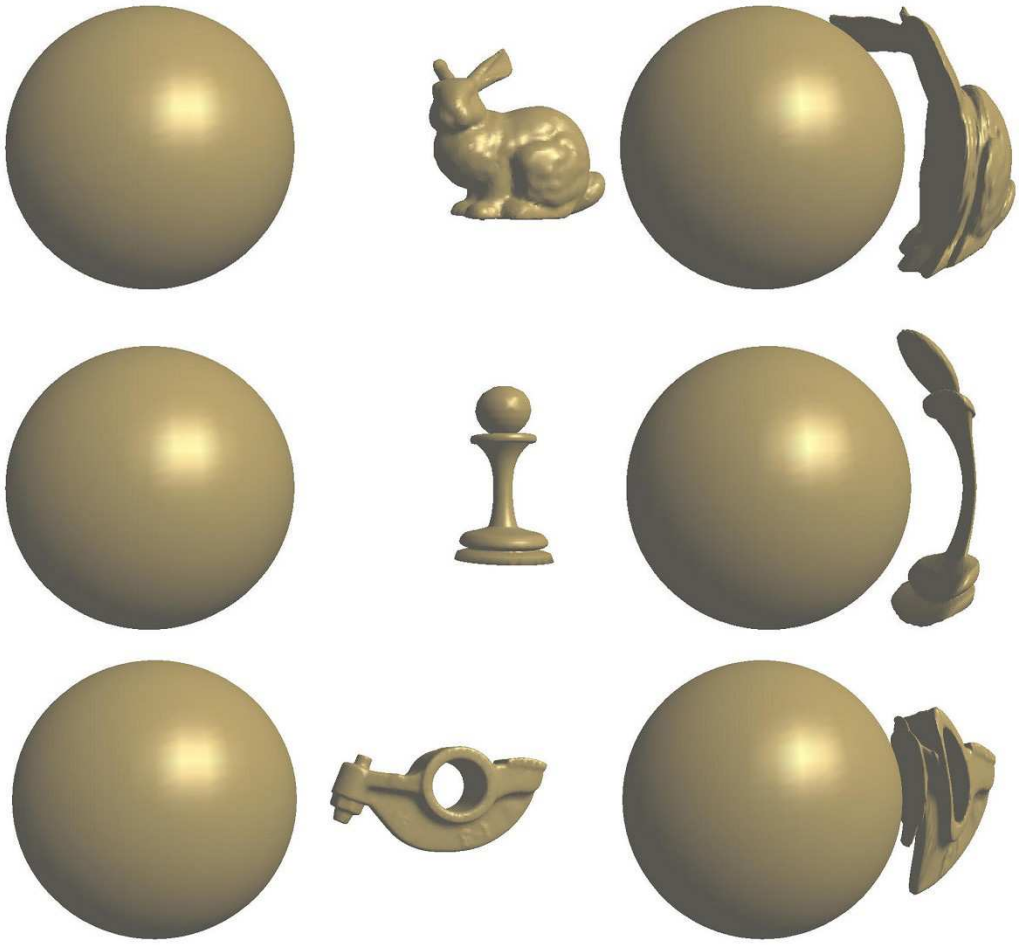


Figure 12.4: Three collision examples: bunny, chess and rocker-arm.

We introduce the distance field $d(x, y, z)$ as one of the components in our vector field construction to represents the shape. The value of such field is the distance to

the object surface. which separates the inner and outer region. Deformed object should only move in the outer region during deformation. Distance field is negative in the inner region when it is positive in the outer region. Obviously, the value of distance field will be zero on the object surface. We want our constructed vector field to affect only the adjacent space of deforming object but not the whole space, so we denote a field threshold $\mathbf{R}$: when absolute distance $d(x, y, z)$ is larger then R, no vector field exists. The blending in base field will insure the smoothness of the curl field.

Therefore, the formulation of the divergence-free vector field for translation $\mathbf{V_T}$ is

$$\begin{cases} \mathbf{V_T}(X) &= \mathbf{curl}(\mathbf{f}(X)) \\ \mathbf{f}(X) &= (R - |dist|)^2 \mathbf{B_T}(X), \quad if|dist| \leq R \\ \mathbf{f}(X) &= 0, \quad if|dist| > R \end{cases} \tag{12.2}$$

Under the definition above, the divergence-free vector field for translation $\mathbf{V_T}$ flows out from the front of the deforming object, and flows along the surface of object and goes back into the object at the back(figure 12.5). Compared with [52], such definition has similar performance for spherical objects but will work better for free-form objects, because our new construction contains shape information, while the vector field in [52] is only defined over a spherical region. [52] uses an array of spherical vector fields to simulate the field for general objects. Our new approach, on the other hand, provides a direct definition for free-form objects such that an array of simple vector field is not necessary.

12.2 Rotation

Assume the deforming object rotates around a pivot o about an **axis**, We can define the curl vector field for rotational motion as follows:

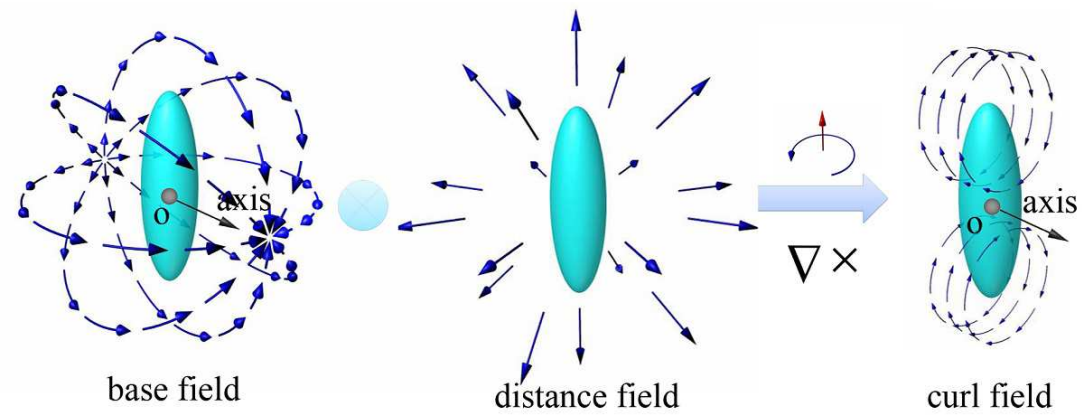


Figure 12.5: Illustration of curl vector field generation for rotational motion. **o** is rotational pivot and **axis** is rotational axis.

$$\begin{cases} \mathbf{V_R}(X) &= \mathbf{curl}(\mathbf{f}(X)) \\ \mathbf{f}(X) &= (R-|dist|)^2\mathbf{B_R}(X), \quad if|dist| \leq R \\ \mathbf{f}(X) &= 0, \quad if|dist| > R \\ \mathbf{B_R}(X) &= X \times \mathbf{axis} \times X \end{cases} \tag{12.3}$$

Figure 12.5 shows how the curl vector field is generated. Figure 12.6 shows the generated rotational field and its deformation effect for a rod-like object with the rotational axis passing through the center.

12.3 Special application: twisting and bending

Besides translation and rotation, we also have special applications with curl vector field construction. These kinds of constructions provide useful effects for object deformation: twist and bending.

Figure 12.7 and equation 12.4 show the construction of simple rotational field that can be used for twisting and bending

$$v(x) = \mathbf{curl}(X \times \mathbf{axis} \times X) \tag{12.4}$$

We construct curl vector field of twisting effect by introducing vector projection

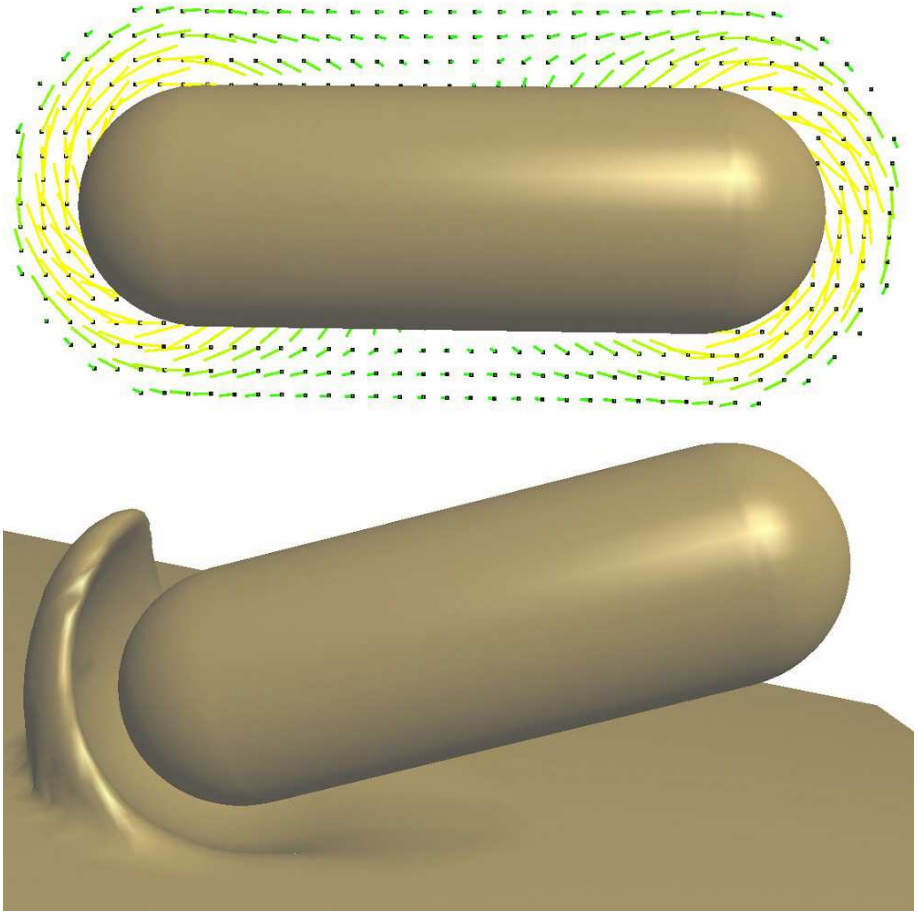


Figure 12.6: The upper figure shows the vectors of the field; the lower figure demonstrates a collision effect between a rotating rod and a surface.

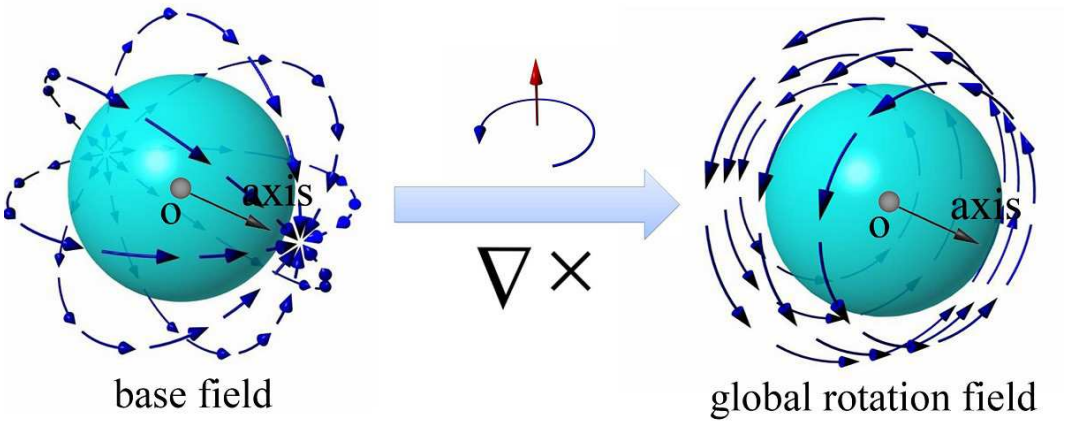


Figure 12.7: Simple rotational field generation.

length L_{axis} on the rotational axis as the constraint parameter in equation 12.4:

$$v(x) = \mathbf{curl}(L_{axis} \cdot X \times \mathbf{axis} \times X) \tag{12.5}$$

Figure 12.8 performs a smooth twisting effect on rectangular rod. Figure 12.9 shows how twisting is applied on the Venus model.

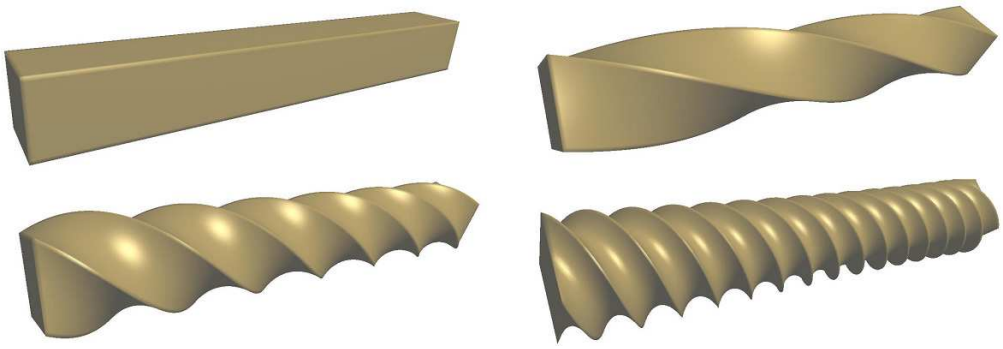


Figure 12.8: Rectangular rod in curl vector field of twist effect.

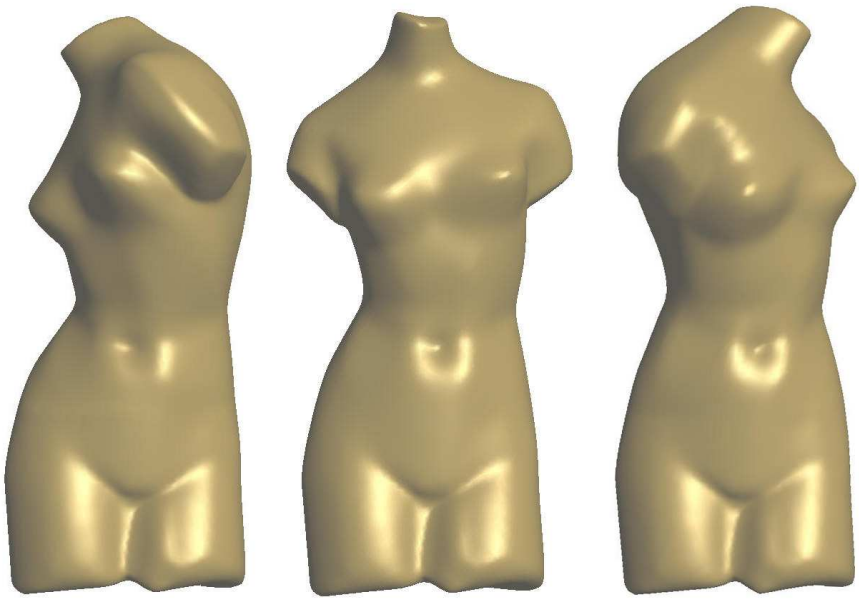


Figure 12.9: Human body action: waist twist of Venus.

Bending effect's construction is similar to twisting effect. We use the square of vector length to be the constraint parameter instead of of the vector projection

length on rotation axis(equation 12.6). Figure 12.10 shows how the bending effect works on a rectangular rod. Figure 12.11 is a Tai Chi image generated by bending effect.

$$v(x) = \mathbf{curl}(|X|^2 \cdot X \times \mathbf{axis} \times X) \tag{12.6}$$

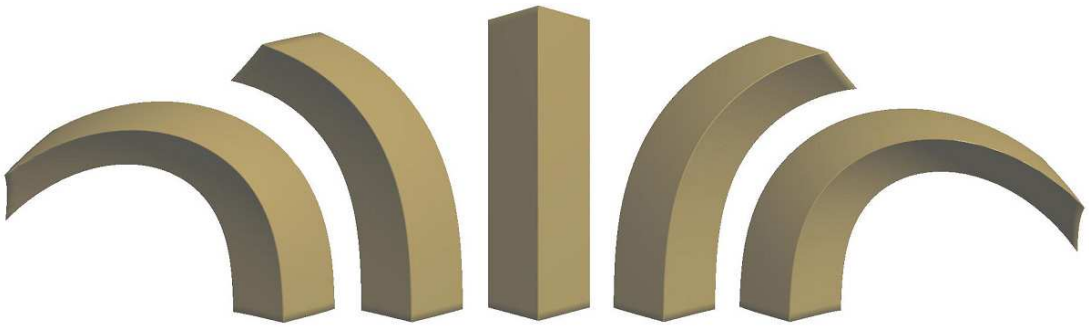


Figure 12.10: Cube in curl vector field of bending effect.

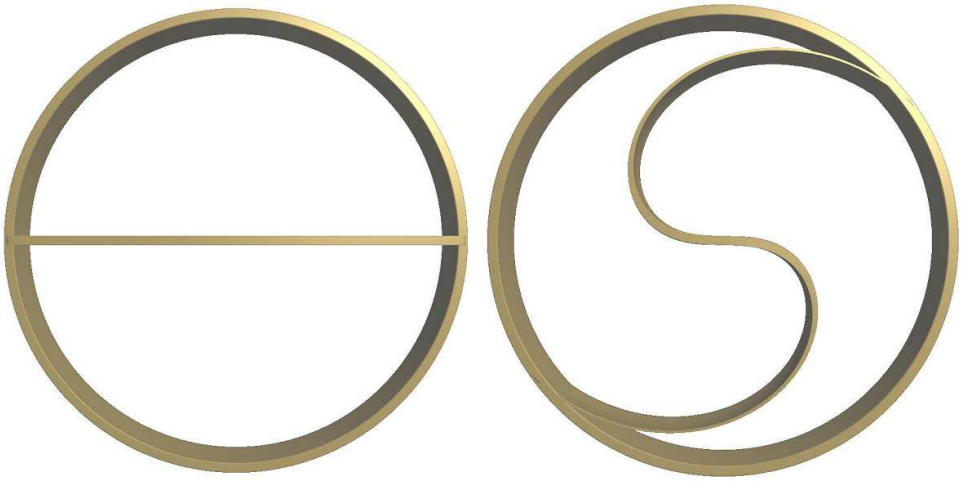


Figure 12.11: Tai Chi generated by curl vector field of bending effect.

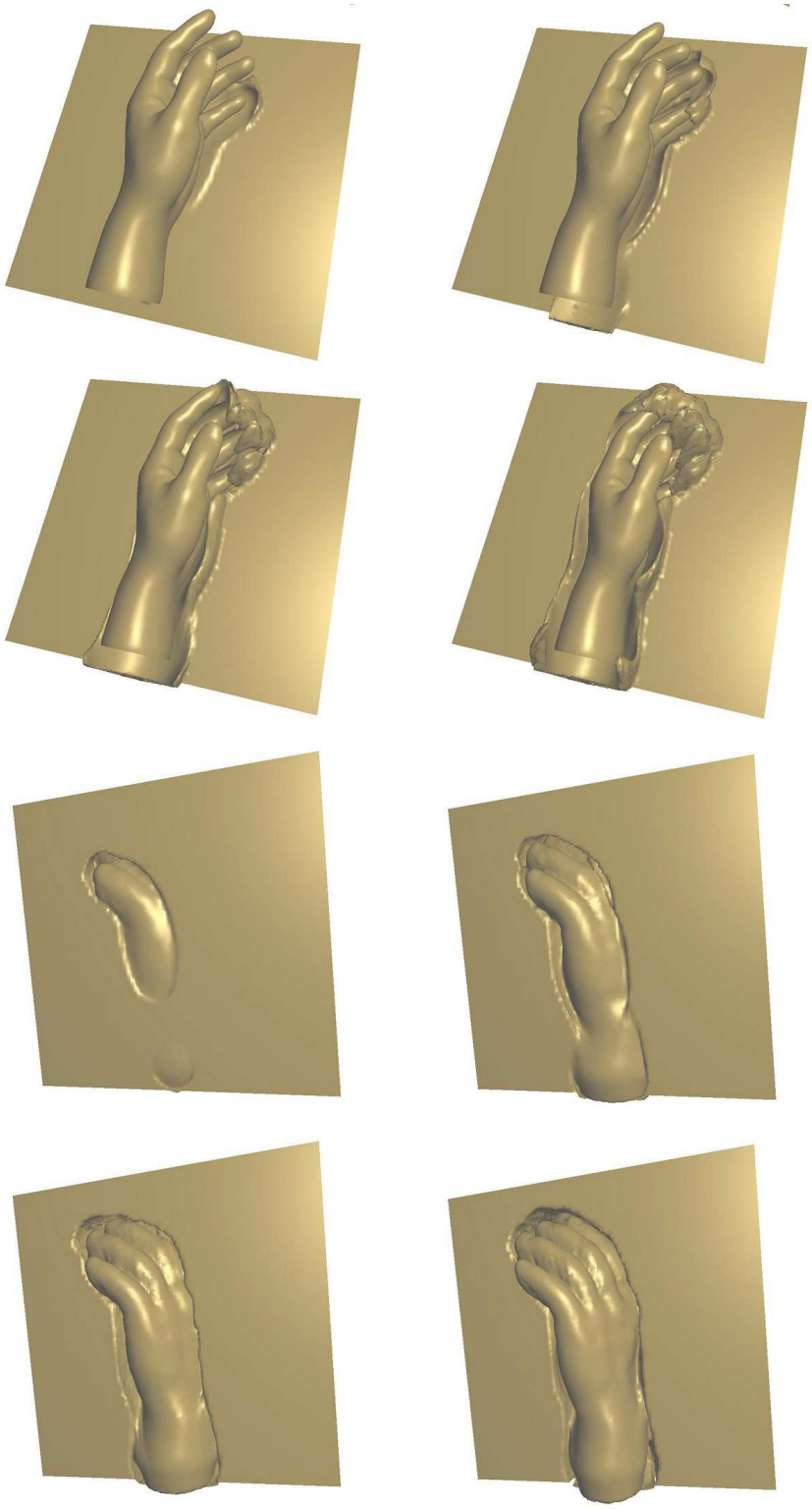


Figure 12.12: Deformation of Complicate objects: hand print on earth.

Chapter 13

Implements and discussion

13.1 Distance field computation

The distance field is an important component in our divergence-free vector field construction. However, distance field computation is time-consuming. To deal with these problems, we use some ideas from [38] to create an implicit function for the domain, which has accurate reconstruction of sharp features and fast local domain access. In order to perform realtime deformation we also use the classic marching cube algorithm [37] with trilinear interpolation to speed up our distance field computation.

13.2 Remeshing

Keeping appropriate vertex density of surface is a basis for feature preserving in deformation. prastic deformation will cause mesh fragmentation and volume change. We apply remeshing in our algorithm based on the ideas from [26] and [52]. We set up several thresholds on triangle edges. When the length of an edge is over the the threshold, a 1-to-4 split will occur at all the ring-triangles of the related vertex or 2 to 4 splits will occur at two nearby triangles of long edges, depending on their topological conditions. We also notices that if we store the original mesh

and deformation path, when ring-triangles of vertices or nearby-triangles of edges require split, we can apply it to the stored original mesh and then integrate the new vertices through deformation path. This operation produces smoother remeshing effect than direct remeshing [52].

	$Tris.$	$volume_o$	$volume_d$	$F.rate$
sphere(C)	11200	2.132	2.127	78
Bonny(C)	23380	1.997	1.988	36
chess(C)	23844	0.576	0.572	36
rocker-arm(C)	43468	0.143	0.142	30
cube(R)	33600	8.000	8.033	39
Venus(R)	36396	74.811	74.324	37

Table 13.1: Column of $Tris.$. records the number of object triangles; column of $volume_o$ and $volume_d$ records the volume data before deformation and after deformation; column of F.rate records the average frame rate in deformation. The object with $'C'$ in bracket was in colliding deformation while the others was in rotating deformation.

13.3 Experiments

All of the test cases were run on a PC with an Intel Xeon 2.66 GHz CPU and 2.00 GB RAM. Our algorithm has smooth performance in realtime without GPU speed-up. Table 13.1 below shows that the volume is preserved under the deformation of curl vector field. Because we use triangular mesh objects in our experiments, there exists tiny error in volume computation. We can see that there is about 0.3% difference between the object volume before and after deformation.

13.4 Advantage and Limitations

Compared with other classical deformation methods, the curl vector field deformation has many obvious advantages. Volume-preserving is an inherent feature of curl vector field, because the field lines of the curl of a based vector field will never intersect with each other, a mesh deformed under a curl vector field can avoid any self-intersection. Compared with the method presented by [52], the new construction is easier to understand, since the construction in [52] requires cross product of two vector components while the new construction is generated from a single curl field directly. However, this vector-field driven deformation also has

some limitations.

13.5 Conclusion and Future Work

In the part, we have presented a brand-new method to construct vector field for 3D mesh deformation. The object deformation under the curl vector field will be volume-preserving, and there is no self-intersection problem in such deformation due to the properties of the vector field. The algorithm introduces distance field into vector field construction, so the shape of the curl vector field is closely related to the object shape. The construction has simple mathematical expression and it is easy to understand. We have defined the construction of the curl vector field for translation and rotation. We also provided some special effects such as twisting and bending. Although our algorithm is fast enough to perform at realtime deformation without GPU speed-up technology, we are also interested in knowing the possibility of implement on GPU.

Bibliography

[1] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2-3):105–114, 2003.

[2] Alexis Angelidis, Marie-Paule Cani, Geoff Wyvill, and Scott King. Swirling sweepers: Constant-volume modeling. In *In Computer Graphics and Applications,12th Pacific Conference on (PG'04),*, pages 10–15, 2004.

[3] D. C. Banks and B. A. Singer. A predictor-corrector technique for visualizing unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):151–163, 1995.

[4] Alan H. Barr. Global and local deformations of solid primitives. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques SIGGRAPH '84*, pages 21 – 30. ACM Press New York, NY, USA, 1984.

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

[6] D. Bechmann and D. Gerber. Arbitrary shaped deformation with dogme. *The Visual Computer*, 19(2-3):175–186, 2003.

[7] R. L. Bishop. There is more than one way to frame a curve. *American Mathematics Monthly*, 82(3):246–251, 1975.

[8] J. Bloomenthal. Modeling the mighty maple. In *Proceedings of SIGGRAPH 1985*, pages 305–311, 1985.

[9] J. Bloomenthal. *Caculation of reference frames along a space curve*. 1990.

[10] M. Bloomenthal and R. F. Riesenfeld. Approximation of sweep surfaces by tensor product NURBS. In *SPIE Proceedings: Curves and Surfaces in Computer Vision and Graphics II*, volume 1610, pages 132–154, 1991.

[11] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. In *International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH 2004*, pages 630 – 634. ACM Press New York, NY, USA, 2004.

[12] Mario Botsch and Leif Kobbelt. Real-time shape editing using radial basis functions. In *Computer Graphics Forum (Proceedings Eurographics 2005)*, volume 24, pages 611–621, 2005.

[13] W. F. Broonsvoort and F. Flok. Ray tracing generalized cylinders. *ACM Transactions on Graphics*, 4(4):291–302, 1985.

[14] H. I. Choi, S-H Kwon, and N-S. Wee. Almost rotation-minimizing rational parametrization of canal surfaces. *Gomputer Aided Geometric Design*, 21(9):859–881, 2004.

[15] T. L. Chung and W. Wang. Discrete moving frames for sweep surface modeling. In *Proceedings of Pacific Graphics'96*, pages 159–173, 1996.

[16] Sabine Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques SIGGRAPH '90*, pages 187 – 196. ACM Press New York, NY, USA, 1990.

[17] H DAVIS. *Introduction to vector analysis*. Allyn and Bacon, Inc., Boston, 1967.

[18] R. Farouki. Exact rotation-minimizing frames for spatial Pythagorean-hodograph curves. *Graphical Models*, 64:382–395, 2002.

[19] R. Farouki and C. Y. Han. Rational approximation schemes for rotation-minimizing frames on Pythagorean-hodograph curves. *Computer Aided Geometric Design*, 20(7):435–454, 2003.

[20] H. W. Guggenheimer. Computing frames along a trajectory. *Gomputer Aided Geometric Design*, 6:77–78, 1989.

[21] A. Hanson. Constrained optimal framing of curves and surfaces using quaternion gauss map. In *Proceedings of Visulization'98*, pages 375–382, 1998.

[22] A. Hanson. *Visualizing Quaternions*. Morgan Kaufmann, 2005.

[23] A. J. Hanson and H. Ma. A quaternion approach to streamline visualization. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):164–174, 1995.

[24] Gentaro Hirota, Renee Maheshwari, and Ming C. Lin. Fast volume-preserving free form deformation using multi-level optimization. In *Proceedings of the fifth ACM symposium on Solid modeling and applications SMA '99*, pages 234 – 245, ACM Press New York, NY, USA, 1999.

[25] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformations. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques SIGGRAPH '92*, pages 177 – 184. ACM Press New York, NY, USA, 1992.

[26] N. James and E. Gain. Adaptive refinement and decimation under free-form deformation. *Proceedings of Eurographics UK '99*, 7(4), 1999.

[27] B. Jüttler. Rotational minimizing spherical motions. In *Advacnes in Robotics: Analysis and Control*, pages 413–422. Kluwer Dordrecht, 1998.

[28] B. Jüttler. Rational approximation of rotation minimizing frames using Pythagorean-hodograph cubics. *Journal of Geometry and Graphics*, 3:141–159, 1999.

[29] B. Jüttler and C. Mäurer. Cubic Pythagorean hodograph spline curves and applications to sweep surface modeling. *Computer-aided Design*, 31:73–83, 1999.

[30] B. Jüttler, M. Peternell, and W. Wang. Möbius invariant frames of space curves. in preparation, 20xx.

[31] F. Klok. Two moving frames for sweeping along a 3D trajectory. *Computer Aided Geometric Design*, 3(1):217–229, 1986.

[32] E. Kreyszig. *Differential Geometry*. Dover, 1991.

[33] F. Lazarus, S. Coquillart, and P. Jancène. Interactive axial deformations. In *Modeling in Computer Graphics*, pages 241–254. Springer Verlag, 1993.

[34] F. Lazarus and A. Verroust. Feature-based shape transformation for polyhedral objects. In *Proceedings of the 5th Eurographics Workshop on Animation and Simulation*, pages 1–14, 1994.

[35] S. Coquillart Lazarus and P. Jancene. Axial deformation: an intuitive technique. *Computer-Aided Design*, 26(8):607–613, 1994.

[36] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, and Christian. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.

[37] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163 – 169. ACM Press New York, NY, USA, 1987.

[38] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *ACM Transactions on Graphics (TOG) , ACM SIGGRAPH 2003 Papers SIGGRAPH '03*, pages 463 – 470. ACM Press New York, NY, USA, 2003.

[39] Q. Peng, X. Jin, and J. Feng. Arc-length-based axial deformation and length preserving deformation. In *Proceedings of Computer Animation 1997*, pages 86–92, 1997.

[40] T. Poston, S. Fang, and W. Lawton. Computing and approximating sweeping surfaces based on rotation minimizing frames. In *Proceedings of the 4-th International Conference on CAD/CG, Wuhan, China*, 1995.

[41] H. Pottmann and M. Wagner. Contributions to motion based surface design. *International Journal of Shape Modeling*, 4(3&4):183–196, 1998.

[42] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques SIGGRAPH '00*, pages 465 – 470. ACM Press/Addison-Wesley Publishing Co, 2000.

[43] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques SIGGRAPH '86*, pages 151 – 160. ACM Press New York, NY, USA, 1986.

[44] S. K. Semwal and J. Hallauer. Biomedical modeling: implementing line-of-action algorithm for human muscles and bones using generalized cylinders. *Computers and Graphics*, 18(1):105–112, 1994.

[45] U. Shani and D. H. Ballard. Splines as embeddings for generalized cylinders. *Computer Vision, Graphics, and Image Processing*, 27:129–156, 1984.

[46] P. Siltanen and C. Woodward. Normal orientation methods for 3D offset curves, sweep surfaces, skinning. In *Proceedings of Eurographics'92*, pages 449–457, 1992.

[47] Karan Singh and Eugene Fiume. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques SIGGRAPH '98*, pages 405 – 414. ACM Press New York, NY, USA, 1998.

[48] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rssl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175 – 184. ACM Press New York, NY, USA, 2004.

[49] Jos Stam. Flows on surfaces of arbitrary topology. In *ACM Transactions on Graphics (TOG) , ACM SIGGRAPH 2003 Papers SIGGRAPH '03*, pages 724 – 731. ACM Press New York, NY, USA, 2003.

[50] Holger Theisel, Tino Weinkauf, Hans-Christian Hege, and Hans-Peter Seidel. Topological methods for 2d time-dependent vector fields based on stream lines and path lines. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):383 – 394, 2005.

[51] J. van Wijk. Image based flow visualization for curved surfaces. In *Proceedings IEEE Visualization*, pages 123–130, 2003.

[52] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. In *International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH 2006 Papers*, pages 1118 – 1125. ACM Press New York, NY, USA, 2006.

[53] W. Wang and B. Joe. Robust computation of rotation minimizing frame for sweep surface modeling. *Gomputer-Aided Design*, 29:379–391, 1997.

[54] E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. *ACM Transactions on Graphics (TOG)*, 25(4):1294 – 1326, 2006.

[55] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. In *ACM Transactions on Graphics (TOG) , ACM SIGGRAPH 2005 Papers SIGGRAPH '05*, pages 496 – 503. ACM Press New York, NY, USA, 2005.