DOI: 10.1111/cgf.15021
EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 2

GLS-PIA: n-Dimensional Spherical B-Spline Curve Fitting based on Geodesic Least Square with Adaptive Knot Placement

Yuming Zhao[1], Zhongke Wu[1†] and Xingce Wang[1]

[1]Beijing Normal University, China

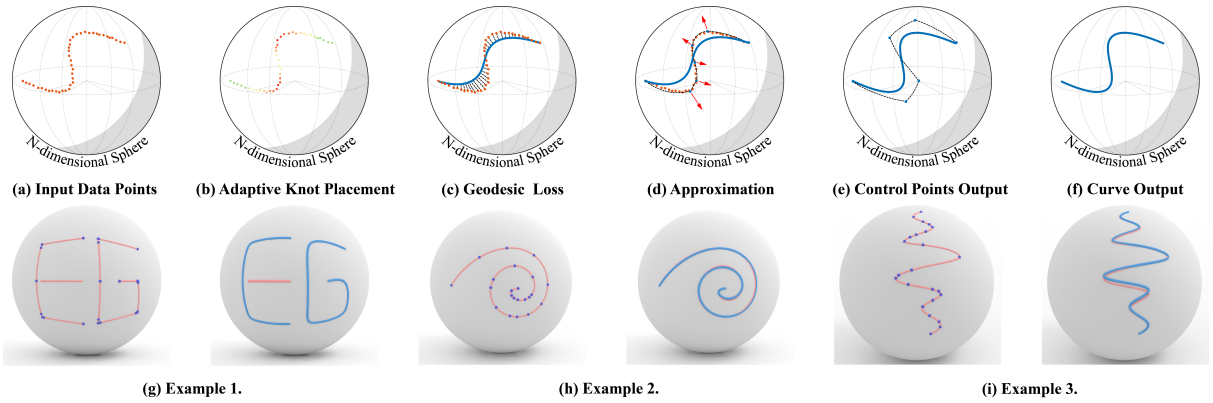


Figure 1: *Main steps of GLS-PIA (Geodesic Least Square-Progressive Iterative Approximation) and examples.*

Abstract

Due to the widespread applications of curves on n-dimensional spheres, fitting curves on n-dimensional spheres has received increasing attention in recent years. However, due to the non-Euclidean nature of spheres, curve fitting methods on n-dimensional spheres often struggle to balance fitting accuracy and curve fairness. In this paper, we propose a new fitting framework, GLS-PIA, for parameterized point sets on n-dimensional spheres to address the challenge. Meanwhile, we provide the proof of the method. Firstly, we propose a progressive iterative approximation method based on geodesic least squares which can directly optimize the geodesic least squares loss on the n-sphere, improving the accuracy of the fitting. Additionally, we use an error allocation method based on contribution coefficients to ensure the fairness of the fitting curve. Secondly, we propose an adaptive knot placement method based on geodesic difference to estimate a more reasonable distribution of control points in the parameter domain, placing more control points in areas with greater detail. This enables B-spline curves to capture more details with a limited number of control points. Experimental results demonstrate that our framework achieves outstanding performance, especially in handling imbalanced data points. (In this paper, "sphere" refers to n-sphere $(n \geq 2)$ unless otherwise specified.)

CCS Concepts

• ***Computing methodologies*** → ***Parametric curve and surface models;***

1. Introduction

B-spline curves on Riemannian manifolds have been receiving increasing attention. Spheres are the most widely used manifold. Data become points on spheres after nomalization. Therefore, B-spline curves on spherical surfaces of different dimensions are essential tools for solving a wide range of problems, including spherical curve modeling, rigid motion description, and temporal data description in shape space [AMAS16, Sho85, HGFL15, SDK*12, KDLS20, HLGQ05]. Specifically, B-spline curves on S^2 are applied in the field of spherical modeling, such as fitting Earth remote sensing data and constructing vector maps of the Earth [AMAS16, AS19]. B-spline curves on S^3 are applied in describing inertial navigation, particularly in the description of rigid

† Corresponding author.

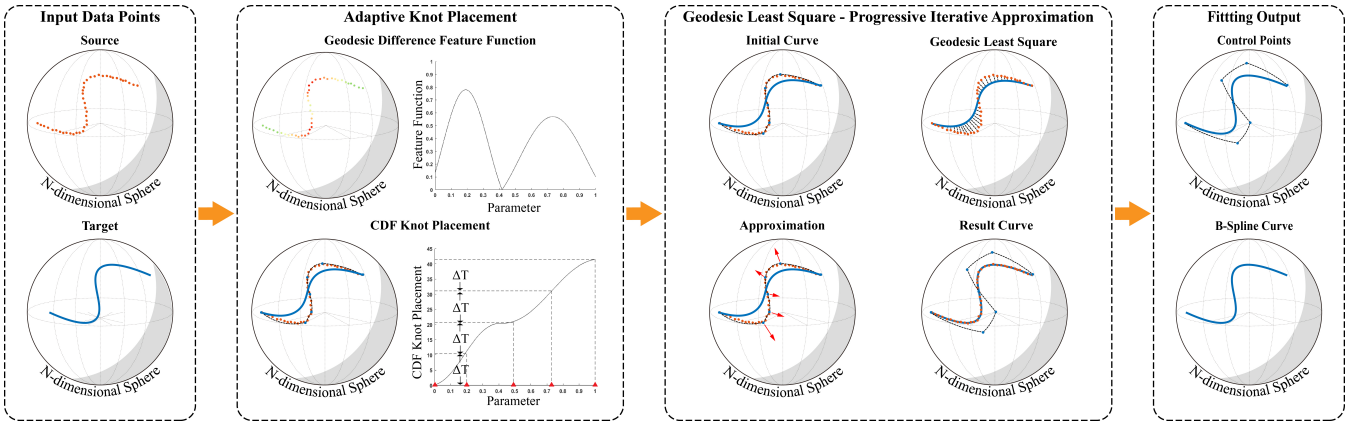


Figure 2: *The pipeline of GLS-PIA with Adaptive Knot Placement. First, the distribution of knot vectors and control points in the parameter domain are estimated based on geodesic difference. Next, the position of the control points are adjusted using GLS-PIA method and the data points are fitted. Finally, the fitted spherical B-spline curve is obtained.*

motion. Rigid motion includes two parts: translation and rotation, and the rotation motion can be described using curves on S^3 [Sho85, HGFL15]. B-spline curves on S^n are used to describe temporal data in shape space [SDK*12]. In the field of shape analysis, shape space is an essential tool. In shape space, each shape is described as a point, and the similarity between two shapes can be calculated by computing the Riemannian metric between two points. Curves in shape space can express temporal data of a shape and can be used to describe shape changes. Common applications include facial expressions, human movements, and protein folding processes [KDLS20].

The problem of fitting a spherical spline curve often needs to balance fitting accuracy and fairness of the curve. Since the spherical manifold is a non-Euclidean space, geometric calculations in non-Euclidean space are often nonlinear, making many effective fitting methods in Euclidean space difficult to directly transfer to the spherical manifold. This poses a great challenge to curve fitting on the n-sphere. Furthermore, commonly used descriptions of fitting accuracy, such as least squares loss, and descriptions of curve fairness, such as strain energy function, are difficult to optimize on the n-sphere. An effective method is to map the fitting problem on the manifold to a Euclidean space, where it is easier to optimize both the fitting accuracy and fairness of the curve. Currently, most methods are based on the tangent space [JK18, KDLS20, GMA18, SASK12], which transforms the fitting problem on the n-sphere to a fitting problem in the tangent space at a specific point on the n-sphere. The tangent space is a Euclidean space, and fitting in the tangent space is straightforward. However, the mapping between the tangent space and the spherical manifold still introduces some errors, which require further optimization to improve the accuracy of the fitting curve, resulting in additional optimization steps and performance overhead.

In this paper, we adopt the geodesic least squares loss as the loss function to accurately characterize the fitting error of the curve on the n-sphere. To address the challenge of directly optimizing geodesic least squares loss on the n-sphere, we leverage the progressive iterative approximation (PIA) approach to optimize it by directly adjusting the control points' positions on the n-sphere according to the error vector. To ensure the smoothness of the fitting curve, we design an error allocation method based on the contribution coefficients of control points, which effectively fits the error with a B-spline curve with fewer control points, thereby ensuring the fairness of the fitted curve. To improve the fitting accuracy of the B-spline curve with a limited number of control points, we need to estimate a more reasonable distribution of knot vectors and control points in the parameter domain, so that they are placed more in the detail-rich areas of the data. We propose an adaptive knot vector placement method based on geodesic difference, using the cumulative distribution function of geodesic difference to characterize the density distribution of details in the input data, and estimate the distribution of knot vectors and control points accordingly. This enhances the B-spline curve's ability to capture data detail information and improves the curve fitting accuracy with a limited number of control points.

Our main contributions are as follows:

- A novel progressive iterative approximation method based on geodesic least squares(GLS-PIA) for fitting B-spline curves on n-dimensional sphere is proposed. By optimizing the geodesic least squares loss through adjusting the control points, GLS-PIA solves the challenge of directly optimizing on the n-sphere. It achieves direct fitting of data points on the n-sphere and improves the accuracy of the curve fitting. Meanwhile, we prove the convergence of the method.
- An adaptive knot placement method based on geodesic difference to enhance the capability of B-spline curves in capturing complex details in data is proposed. Geodesic difference is designed to characterize the geometric features of input discrete points and a feature function based on the cumulative distribution function (CDF) is used to describe the detail density. Higher-density knots are placed in regions with larger feature values so that the fitted curve can better capture the geometric details of the input points. Moreover, this method is computationally efficient and does not require iterative optimization of the knot vector.
- An error allocation method based on contribution coefficients is

proposed. This method effectively reduces the number of control points during the fitting process, thus maintaining the smoothness of the curve.

2. Related Work

In this section, we will review the relevant work on the construction and fitting methods of spherical B-spline curves.

2.1. Spline Curve Construction on n-Sphere

In Euclidean space, there are many classical spline curve representations, such as Bézier curves, B-spline curves, and non-uniform rational B-spline (NURBS) curves. However, due to the non-Euclidean nature of spherical space, these classical spline representations are difficult to directly introduce into the n-sphere manifold. Based on these methods, two types of spline curve construction methods on the n-sphere have been developed.

Projection method is the first kind. A curve defined in the R^{n+1} space is projected onto the S^n. In 1979, Parker [PD79] proposed a simple method by constructing a curve in R^3 and then projecting it onto the sphere by normalization. Although this method is simple and easy to implement, it can cause significant geometric distortion. In 1995, Kim [KKS95b, KKS95a] proposed a method that transforms Bézier and Hermite curves into similar unit quaternion curves using exponential mapping [KKS96] and quaternion operations [KN95], which iteratively solves a nonlinear equation system of quaternions to obtain the position information of control points, thus better solving the interpolation problem of spherical curves. Based on this work, Tan [TXFH18] fit the corresponding derivative loss with several adjacent interpolation points, and then calculate the control points, thus constructing the curve. However, this kind of methods has certain limitations, as they usually construct the curve before projection, resulting in insufficient accuracy, geometric distortion, or complicated and inefficient computation.

Direct construction method is the second kind, which directly constructs spline curves on the n-sphere. In 1985, Shoemake [Sho85] first introduced the concept of quaternions into computer graphics and constructed a new type of curve, which is a unit quaternion spline curve satisfying C^1 continuity. When constructing this curve, Shoemake replaced line segments with great circle arcs (i.e., the shortest geodesics in Riemannian geometry) and extended the De Casteljau algorithm in Euclidean space to low-dimensional unit spheres, and gave the conditions for C^1 smooth splicing of two spherical Bézier curves. Although Shoemake only considered cubic Bézier curves in Euclidean space, this method can be extended to spheres of any dimension. Furthermore, Popiel [PN06] studied the properties of spherical Bézier curves of any dimension at the endpoints and constructed C^2 spherical Bézier curves using the concept of conjugate derivatives. This method solves the C^2 interpolation problem of data points and significantly reduces the complexity of calculating high-order derivatives at the endpoints of spherical Bézier curves, but still cannot achieve local control of curve shape. Crouch extended the De Casteljau algorithm to S^n manifolds and Lie groups [CKL99]. Huang used the method of knot insertion with De Boor to calculate NURBS curves on the sphere [HGFL15]. This method directly utilizes the properties of the sphere manifold and achieves direct and efficient spherical spline construction computation. Furthermore, Cao introduced the Spherical DCB-Spline as a means to construct a non-tensor-product B-spline on the surface of a sphere [CLCQ12].

Overall, the projection method has the advantages of generality, as the method of projecting curves from Euclidean space to manifolds is general and easy to extend to general manifolds. However, it fails to fully utilize the special properties of spherical manifolds, resulting in lower computational efficiency and greater geometric distortion. The direct construction method fully utilizes the properties of spherical manifolds and extends the methods of constructing curves in Euclidean space to spherical manifolds, with the advantages of high efficiency and less geometric distortion. However, this kind of methods also have the disadvantage of being difficult to extend to general manifolds. It can be seen that for spherical curve methods, the direct construction method is a more optimal method to focus on.

2.2. B-Spline Curve Fitting on n-Sphere

Methods for fitting on manifolds involve mapping the data points to a vector space, usually the tangent space at a point on the manifold, computing the fitting curve on the tangent space, and then mapping the resulting curve back to the manifold. The mapping between the manifold and its tangent space can be defined, for example, using a rolling procedure [JK18,SHSH07,Pre87,KKL10,Uk07,MKM*99]. For instance, Jupp and Kent [JK18] proposed a method that can be described as "unwrapping" the data onto the plane, where standard curve fitting techniques can then be applied. The papers [KKL10, MKM*99] studied the problem of fitting a smooth curve on the planar shape space. Gousenbourger proposed a method for fitting composite Bézier-like curves and blended cubic splines on the sphere using the exponential map and logarithmic map [GMA18]. Samir provides an expression for the gradient vector on a general manifold in terms of its differential geometry [SASK12].

There are also methods that do not rely on tangent spaces to handle data on manifolds. Alderson proposed a curve multiresolution [AMAS16] and multiscale [AS19] method on the sphere and ellipsoid surfaces. These methods can be indirectly used for fitting discrete points on the sphere. Such methods are based on curve subdivision techniques, which have the advantage of avoiding errors introduced by the mapping between tangent spaces and the sphere. However, these methods can only control the curve reconstruction effect by adjusting the parameters of curve subdivision, and they lack explicit control over the error between the discrete points and the resulting curve. Our goal is to develop a method that not only avoids mapping errors between tangent spaces and the sphere, but also explicitly controls the fitting error, in order to easily obtain the best fitting curve.

Most of the current fitting methods rely on the mapping between the tangent space and the manifold. Due to the non-Euclidean nature of the n-sphere, there is always some error in the mapping between the tangent space and the manifold, which can lead to reduced fitting accuracy and slower fitting speed.

The method proposed in this paper directly adjusts the control

points and fits the curve on the n-sphere, without relying on the tangent space for curve fitting. We have designed a series of methods to avoid the use of the tangent space and improve the accuracy of curve fitting. We will introduce the specific technical details later in the paper.

3. Construction of B-Spline Curves on n-Sphere

Considering the non-Euclidean nature of the n-sphere, we will use a spherical B-spline curve method based on spherical interpolation. In this section, we will first introduce the definition of spherical interpolation. Secondly, we will introduce the construction of spherical B-spline curves based on the de Boor algorithm.

3.1. Spherical Linear Interpolation

Two-point interpolation is atomic operation in spherical space that is analogous to the simplest atomic operations used to create curves in Euclidean space. Spherical linear interpolation (SLERP) can be defined by

$$SLERP(p,q,u) = \frac{sin[(1-u)\theta]}{sin(\theta)}p + \frac{sin(u\theta)}{sin(\theta)}q \tag{1}$$

p and q are two points on unit n-sphere, θ is the angle between p and q, can be calculated by

$$\theta = cos^{-1}(p \cdot q) \tag{2}$$

With the SLERP, We can extend the de Boor algorithm from Euclidean space to the n-sphere.

3.2. De Boor's Algorithm on n-Sphere

We use de Boor algorithm to calculate B-spline curves on the n-sphere. Given a set of control points $P_i(i = 0,1,2...)$ on the unit n-sphere S, the total number of control points is $n+1$, p denotes the degree of B-spline curve, and $U = \{u_0,...,u_p,u_{p+1},...,u_{m-p-1},u_{m-p},...,u_m\}$ denotes knot vector. we can insert a new knot t into U. Assuming t is in the interval $[u_k,u_{k+1})$, after r insertions of t, the new control points can be expressed as follows:

$$\begin{aligned}\mathbf{P}_{i,r} &= SLERP(\mathbf{P}_{i-1,r-1},\mathbf{P}_{i,r-1},\alpha_{i,r}) \\ &= \frac{sin[(1-\alpha_{i,r})\theta_{i,r-1}]}{sin\theta_{i,r-1}}\mathbf{P}_{i-1,r-1} + \frac{sin(\alpha_{i,r}\theta_{i,r-1})}{sin\theta_{i,r-1}}\mathbf{P}_{i,r-1}\end{aligned} \tag{3}$$

where

$$\alpha_{i,r} = \begin{cases} 1, & i \le k-p+r-1 \\ \frac{t-u_i}{u_{i+p-r+1}-u_i}, & k-p+r \le i \le k \\ 0, & i \ge k+1 \end{cases} \tag{4}$$

$$\theta_{i,r-1} = cos^{-1}(\mathbf{P}_{i-1,r-1} \cdot \mathbf{P}_{i,r-1}) \tag{5}$$

According to the knot insertion algorithm of B-spline, after inserting t with $r = p$ times, the new point just denotes the point on the B-spline curve with parameter t.

4. Adaptive Knot Placement based on Geodesic Difference

Our motivation stems from the fundamental idea of knot placement method in B-spline curve fitting in Euclidean space [YNPT20, LZJ*17, TDH15, CMRS01, AEC*18, XQ01]. These knot placement methods in Euclidean space have demonstrated excellent results. In particular, fast knot placement [YNPT20] eliminates the need for cumbersome optimizations. Instead, it provides excellent initialization effects for fitting through simple estimation, resulting in a high-speed performance. Inspired by these methods, we have extended them to spherical manifolds to enhance the capability of capturing fine details in curve fitting on the sphere. For a p-order B-spline curve, its $(p-1)$th derivative has discontinuous points that correspond to the knot locations. Given a set of sampled points from a p-order B-spline, the original knot locations can be recovered by locating the discontinuous points in the $(p-1)$th derivative. In practical applications, the input data points are usually not sampled from the B-spline and the derivatives may not exhibit obvious discontinuities. Nonetheless, previous works have shown that the input data differences can be used to guide knot placement, resulting in knots that are consistent with the properties of the input data, and thus better capturing the details of the data.

For points on S^n manifold, a simple way to define differences is to embed the S^n manifold in the Euclidean space R^{n+1} and use Euclidean differences. However, this definition does not eliminate the influence of the non-Euclidean properties of the S^n manifold. For example, for geodesics on the spherical manifold, which correspond to straight lines in Euclidean space, their curvature should be zero. That is, for any three ordered points on the same great circle on the n-sphere, their second-order difference should be zero. However, if Euclidean differences are used in R^{n+1} space, the calculated second-order difference is obviously not zero. To eliminate the influence of the non-Euclidean properties of the S^n manifold, we propose a geodesic difference calculation method to eliminate the curvature of the manifold itself in the tangent space.

In this section, we introduce an adaptive knot placement method based on geodesic differences. First, we define and calculate geodesic differences. Second, we introduce the calculations of the feature function defined by geodesic differences. Finally, we present a knot placement and control point placement method based on the feature function.

4.1. Definition of Geodesic Difference

With a given set of m input points $q = \{q_I : Q_I \in S^n\}_{i=1}^m$ and parameters $U = \{u_i : u_i \in R, u_i < u_{i+1}\}_{i=1}^m$, we define $Q^{(k)} = \{q_j^{(k)} \in S^n\}_{j=1}^{m-k}$ to be the set of kth differences of the input points at parameters $U^{(k)} = \{u_j^{(k)} \in R\}_{j=1}^{m-k}$. We let $Q^{(0)} = Q$, $U^{(0)} = U$, then for $K > 0$, we use the geodesic difference to find q_j^p.

$$q_j^{(k+1)} = \frac{\text{proj}_{T_h S^n} q_{j+1}^{(k)} - \text{proj}_{T_h S^n} q_j^{(k)}}{u_{j+1}^{(k)} - u_j^{(k)}} \tag{6}$$

$$u_j^{k+1} = \frac{u_j^k + u_{j+1}^k}{2} \tag{7}$$

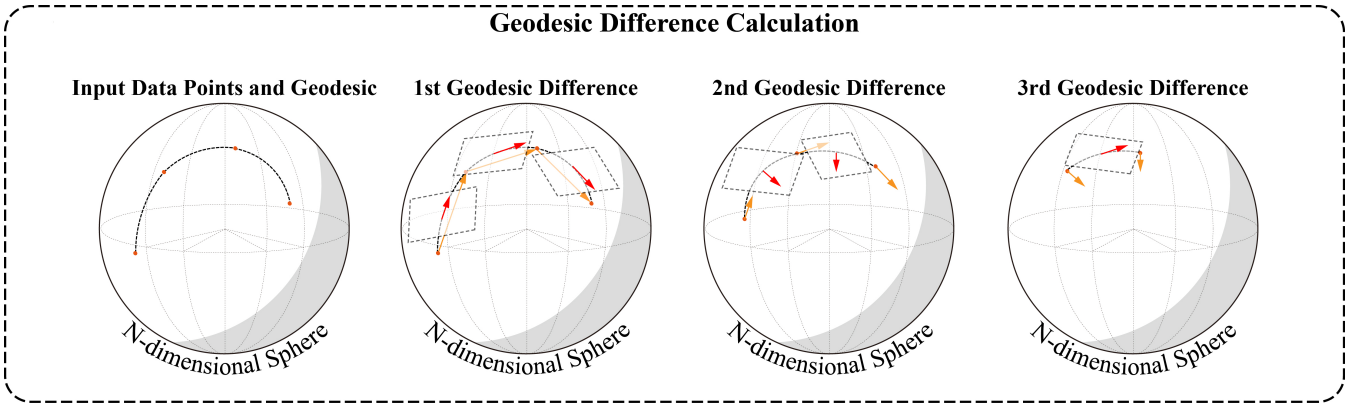


Figure 3: *Geodesic Difference Calculation*

where

$$\text{proj}_{T_h S^n} q_j^{(k)} = q_j^{(k)} - \frac{q_j^{(k)} \cdot h}{|h|^2} h \tag{8}$$

$$h = \frac{q_{j+1}^{(k)} + q_j^{(k)}}{2} \tag{9}$$

h denotes the midpoint of $q_{j+1}^{(k)}$ and $q_j^{(k)}$, $T_h S^n$ denotes the tangent space on h, $\text{proj}_{T_h S^n} q_j^{(k)}$ denotes the projection vector of $q_j^{(k)}$ onto the tangent space of point h.

To mitigate the impact of the manifold's curvature on the curve, we design geodesic difference to characterize the details of the data. During the process of calculating the differentiation, we project the vectors onto the tangent space of the midpoint and perform the difference computation within the tangent space. It is worth noting that when computing the 1st difference, we treat the points as vectors pointing from the center of the n-sphere to the point, and project this vector onto the tangent space of the midpoint for difference computation. This calculation is equivalent to projecting the point onto the tangent space and then performing the difference computation. After this step, all operations are performed on the vector obtained from the previous difference computation. Similarly, when computing the projection, the normal vector of the tangent space is required. Due to the special properties of the unit n-sphere, the vector from the origin of the n-sphere to point h is the unit normal vector of the tangent space at point h. This greatly facilitates our calculations.

4.2. Cumulative Distribution Function(CDF) as Feature Function

We use the cumulative distribution function (CDF) of the dth difference of the data points as a representation of the level of detail in the input data, where d is the degree of the fitted B-spline curve. First, to avoid an excessive concentration of knots in detail-rich areas, we employ a normalization function to smooth the distribution of the difference. The normalization function f uses the pth root of the magnitude. As a result, we obtain the normalized difference, which can be expressed as follows:

$$(u_i^{'}, f_i) = \begin{cases} (u_1, 0), & i = 0 \\ (u_i^d, (\|q_i^{(d)}\|_2)^{1/p}), & 1 \le i \le m-d \\ (u_m, 0), & i = m-d+1 \end{cases} \tag{10}$$

Before computing the CDF, we need to first estimate a continuous difference function from the discrete normalized differences. Here, we obtain the continuous difference function by linearly interpolating the normalized differences over the parameter domain.

$$f(u) = \frac{u - u_{i+1}^{'}}{u_i^{'} - u_{i+1}^{'}} f_i + \frac{u - u_i^{'}}{u_{i+1}^{'} - u_i^{'}} f_{i+1} \tag{11}$$

where $u_i^{'} \le u \le u_{i+1}^{'}$, and $f(u) = 0$ for u outside of the range $[u_0^{'}, u_{m-d+1}^{'}]$.

Finally, we integrate $f(u)$ to obtain the CDF function of the difference, which is used to measure the density distribution of the level of detail in the input data.

$$F(u) = \int_{-\infty}^{u} f(v) dv \tag{12}$$

4.3. Knot Placement and Control Points Placement

We determine the placement of knots and control points based on the Cumulative Distribution Function (CDF). The CDF represents the amount of detail in the input data, and we aim to achieve equal amounts of detail in each interval between two knots. This is achieved by making the CDF difference between the upper and lower bounds of each interval equal. If the number of control points of a B-spline curve is n and its degree is p, then the knot vector is defined as follows:

$$T = \{\overbrace{t_1, ..., t_1}^{p+1}, t_2, ..., t_{n-p}, \overbrace{t_{n-p+1}, ..., t_{n-p+1}}^{p+1}\} \tag{13}$$

Here, t_i is obtained by uniformly sampling the interval $[0, F_{max}]$ and performing $n-p+1$ uniform samplings for a B-spline curve of degree p and n control points. Since F is a cumulative distribution

function of the non-negative f , F is non-decreasing. So we can define F^{-1} as the inverse of F

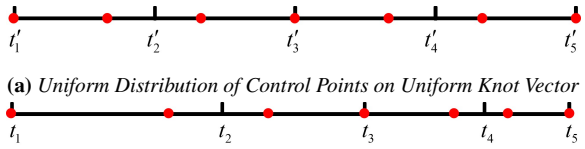
$$F^{-1}(a) = b \Leftrightarrow F(b) = a \tag{14}$$

Furthermore, we can calculate the values of the knot vector by defining the length of the intervals for the F-domain uniform sampling.

$$\Delta F = \frac{F_{max}}{n-p+1} \tag{15}$$

$$F_i = (i-1)\Delta F \tag{16}$$

$$t_i = F^{-1}(F_i) \tag{17}$$



(a) *Uniform Distribution of Control Points on Uniform Knot Vector*

(b) *Distribution of Control Points on Real Knot Vector*

Figure 4: *Control Point Placement Process*

Thus, we obtain the knot vectors, and based on these knots, we provide an initial distribution for the control points. Initially, we assume that all knot values are uniformly distributed in the parameter domain and all control points are uniformly placed. Subsequently, we use a linear transformation to convert the uniform knot values to their actual values and, at the same time, transform the control points to their new positions based on the knot interval in which they belong. This is illustrated in Figure 4.

5. GLS-PIA: Geodesic Least Square - Progressive Iterative Approximation

Due to the non-Euclidean nature of the n-sphere, traditional optimization methods are difficult to use on the n-sphere. PIA provides a B-spline curve approximation method that can be applied on the n-sphere. However, the traditional PIA method initializes all data points as control points, which leads to a serious lack of fairness in the curve generated by the traditional PIA method for large-scale data. To solve this problem, we have designed a PIA method based on geodesic least squares. We use geodesic distance as the measure of fitting error and allocate the total error to each control point according to its contribution coefficient in the B-spline curve, for adjustment.

Following this idea, our approach for GLS-PIA for B-spline curve fitting is comprised of the following steps:

1.Initialize control points based on the adaptive knot placement method.

2.Generate the initial fitting curve.

3.Obtain the adjustment vector based on the geodesic distance error.

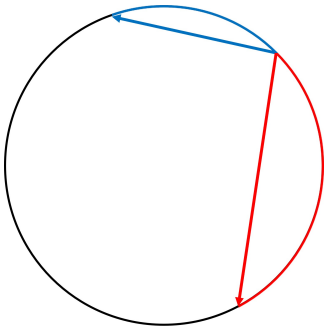


Figure 5: *The difference between chord length and geodesic distance in the red segment is significantly larger than that in the blue segment. If we use chord length as the magnitude of the error vector, it would cause the updates provided by points farther from the curve to be insufficient, leading to a loss of accuracy in fitting the curve.*

4.Allocate the adjustment vector based on the contribution coefficient of each control point.

5.Obtain the new curve.

5.1. Geodesic Least Square and Errors Allocation

For the problem of curve fitting on a spherical manifold, the commonly used error measurement is the geodesic least squares loss, defined as follows:

$$Loss(Q = \{q_i : q_i \in S^n\}_{i=1}^m, C(t)) = \sum_{i=1}^{m} cos^{-1}(q_i \cdot C(u_i)) \tag{18}$$

Directly optimizing this function on the n-sphere is challenging. Typically, the approach is to optimize it in Euclidean space through a mapping between the tangent space and the n-sphere. However, this approach introduces errors. Based on this, we propose a new GLS-PIA method, which indirectly optimizes the geodesic least squares loss by adjusting the control points on the n-sphere.

To facilitate the subsequent PIA optimization and control point adjustment, we need to calculate the error vector from each data point to the corresponding point on the curve. It is defined as follows:

$$\delta_j = \frac{q_j - \mathbf{C}(t_j)}{\|q_j - \mathbf{C}(t_j)\|} cos^{-1}[q_j \cdot \mathbf{C}(t_j)], j = 0, 1, ..., m \tag{19}$$

We use the direction of the chord vector from the corresponding point on the curve to the data point as the direction of the error vector, and the geodesic distance as the magnitude of the error vector. The reason for using geodesic distance as the magnitude of the error vector instead of the length of chord is that on the n-sphere, the difference between chord length and geodesic distance becomes greater as the distance between two points increases.(shown in Figure 5) If we use chord length as the magnitude of the error vector, it would cause the updates provided by points farther from the curve to be insufficient, leading to a loss of accuracy. in fitting the curve.

In traditional PIA, all data points are initialized as control points,

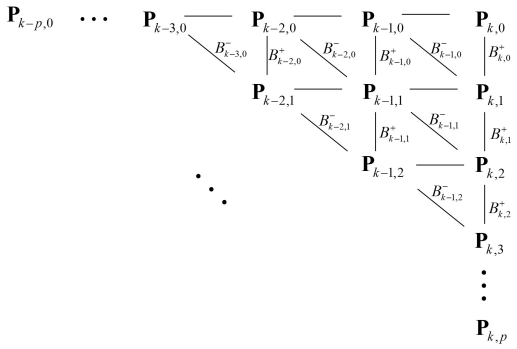


Figure 6: *Calculation of Contribution Coefficient*

allowing fitting errors to be conveniently allocated to each control point through point-to-point correspondence, thereby facilitating the derivation of adjustment vectors. However, this approach has the disadvantage of having too many control points. When dealing with large datasets, it can lead to poor curve fairness. Through the adaptive knot placement method discussed earlier, we have determined the initialization positions of the control points and improved curve fairness by reducing the number of control points.

Regarding the allocation of errors, we determine it through the contribution coefficients of each control point. Let $\mathbf{P}$ be the input data point set, and $\mathbf{U}$ be the parameters. For a point p_i with parameter u_i, we calculate the distance from the point on the curve with parameter u_i to point p_i as its fitting loss. We use the contribution coefficient of each control point to determine the allocation of errors for each point with parameter u_i on the curve. The contribution coefficient is similar to the basis function of a B-spline curve in Euclidean space, with the difference being that the basis function of a B-spline curve in Euclidean space can be calculated independently, while on the n-sphere, due to non-Euclidean properties, the contribution coefficient depends on the intermediate value of the control points calculated using the de Boor algorithm.

The calculation of the contribution coefficient can be achieved using the de Boor algorithm introduced in Chapter 3, as illustrated in the Figure 6.

$B^+_{i,r}$ and $B^-_{i,r}$ is defined as follows:

$$B^-_{i,r} = \frac{sin[(1-\alpha_{i+1,r+1})\theta_{i+1,r}]}{sin\theta_{i+1,r}} \tag{20}$$

$$B^+_{i,r} = \frac{sin(\alpha_{i,r+1}\theta_{i,r})}{sin\theta_{i,r}} \tag{21}$$

To calculate the contribution coefficient of any control point to a point on the curve, we only need to follow the de Boor algorithm and calculate the sum of the products of all $B^{+/-}_{i,r}$ along the paths from that control point to $P_{k,p}$.

With the help of contribution coefficient, we can allocate errors to each control points as follows:

$$\Delta_i = \mu \sum_{j=0}^{m} D_i(t_j)\delta_j \tag{22}$$

Δ_i is the adjustment vector for the control point P_i, μ is the adjusting rate, δ_j denotes the fitting error on point $C(t_j)$. D_i is the contribution coefficient of the control point P_i, which can be calculated by de Boor algorithm with $B^+_{i,r}$ and $B^-_{i,r}$ as illustrated in the Figure 6.

5.2. Control Points Update

After obtaining the adjustment vectors, we can update each control point as follows:

$$P_i^{'} = \frac{P_i + \Delta_i}{\|P_i + \Delta_i\|} \tag{23}$$

$P_i^{'}$ denotes the new control point, $\mathbf{C}(t_j)^{'}$ can be constructed from new control points. To make sure that the new control point $P_i^{'}$ is still lie on the unit n-sphere, we normalize the $P_i^{'}$ by dividing it by its magnitudes.

5.3. Algorithm Framework

Combining the adaptive knot placement method from Section 4 with the GLS-PIA method from Section 5, we obtain the complete algorithm for our framework. For the input data points and their parameters, we first estimate the distribution of the knot vector based on geodesic difference and place the knot vector and control points accordingly. Next, we adjust the position of the control points using the GLS-PIA method and perform spherical B-spline curve fitting for the data points. The detailed algorithm is presented in Algorithm 1.

5.4. Proof of Convergence

To substantiate the effectiveness of our method, in this section, we will establish the convergence of the GLS-PIA method.

Proof Let k represent the iteration number in the GLS-PIA method, Assuming P^k represents the control points obtained at the k-th iteration, and P^0 represents the control points initially provided. According to equation (23), we can express the iteration of control points as follows:

$$P_i^{k+1} = \frac{P_i^k + \Delta_i^k}{\|P_i^k + \Delta_i^k\|} \tag{24}$$

Let m_i^k represent the $\frac{1}{\|P_i^k+\Delta_i^k\|}$. It becomes apparent that m_i^k represents a constant bounded by the interval $(1, \pi/2)$. Consequently, the given equation can be elegantly rephrased as follows:

$$P_i^{k+1} = (P_i^k + \Delta_i^k) * m_i^k \tag{25}$$

In matrix form, the equation can be expressed as follows:

$$\mathbf{P}^{k+1} = (\mathbf{P}^k + \Delta^k) * M^k \tag{26}$$

Algorithm 1: GLS-PIA: Geodesic Least Square-Progressive Iterative Approximation

Input: Input points $Q = \{q_i : q_i \in S^n\}_{i=1}^m$, the parameter $U = \{u_i : u_i \in R, u_i < u_{i+1}\}_{i=1}^m$, error threshold E, number of control points N

Output: A Fitting Spherical B-Spline Curve $\mathbf{C}(t)$ with minimum Geodesic Error

Result: Write here the result

$Error = \infty$;
Calculate geodesic difference of input points;
Normalize geodesic difference by pth root of the magnitude;
Calculate CDF as feature function of geodesic difference;
Place the knot vector and control points by CDF;
Construct the initial curve $\mathbf{C}(t)$;
while $Error \geq E$ **do**
    Calculate the fitting error on point $\mathbf{C}(t_j)$ according to equation (19);
    Calculate the contribution coefficient of each control point;
    Allocate the errors to each control point according to equation (22);
    Update the control points according to equation (23);
    $Error = \sum_{i=1}^m cos^{-1}[q_i \cdot \mathbf{C}(u_i)]$;
end
The last $\mathbf{C}(t)$ is the result Spherical B-spline Curve;

Let A denote the contribution coefficient matrix.

$$A = \begin{bmatrix} D_0(t_0) & D_1(t_0) & \cdots & D_n(t_0) \\ D_0(t_1) & D_1(t_1) & \cdots & D_n(t_1) \\ \vdots & & \ddots & \vdots \\ D_0(t_n) & D_1(t_n) & \cdots & D_n(t_n) \end{bmatrix} \tag{27}$$

By considering equation (19), we can express $\frac{cos^{-1}[q_j \cdot \mathbf{C}(t_j)]}{\|q_j - \mathbf{C}(t_j)\|}$ as a diagonal matrix for the sake of facilitating subsequent processing and analysis, which we will refer to as F:

$$F = \begin{bmatrix} \frac{cos^{-1}[q_0 \cdot \mathbf{C}(t_0)]}{\|q_0 - \mathbf{C}(t_0)\|} & 0 & \cdots & 0 \\ 0 & \frac{cos^{-1}[q_1 \cdot \mathbf{C}(t_1)]}{\|q_1 - \mathbf{C}(t_1)\|} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{cos^{-1}[q_n \cdot \mathbf{C}(t_n)]}{\|q_n - \mathbf{C}(t_n)\|} \end{bmatrix} \tag{28}$$

Expanding equation (26), we obtain the following expression:

$$\mathbf{P}^{k+1} = [\mathbf{P}^k + \mu F * A^T (Q - A\mathbf{P}^k)] * M^k \tag{29}$$

Upon observing F, it is evident that each element within F is bounded. This implies that F does not affect convergence. We can always find a suitable μ to eliminate the influence of F, thus ensuring convergence. Therefore, in the subsequent proof, we exclude F to simplify the process. It is important to reiterate that this approach does not impact the conclusion regarding convergence.

By rewriting it in iterative form, we can obtain the following expression:

$$\begin{aligned} &\mathbf{P}^{k+1} - (M^k + M^k \mu A^T A - I)^{-1}(M^k \mu A^T Q) \\ &= (1 - \mu A^T A)[\mathbf{P}^k - (M^k + M^k \mu A^T A - I)^{-1}(M^k \mu A^T Q)] \\ &= \cdots \\ &= (1 - \mu A^T A)^{k+1}[\mathbf{P}^0 - (M^0 + M^0 \mu A^T A - I)^{-1}(M^0 \mu A^T Q)] \end{aligned} \tag{30}$$

Let E denote the matrix $1 - \mu A^T A$, $\lambda_i(E)$ denote the eigenvalue of E. It is evident that the matrix $A^T A$ is non-singular and positive definite. Given that the maximum eigenvalue of $A^T A$ is λ_0, when $0 < \mu < \frac{2}{\lambda_0}$, we can draw the following conclusion:

$$-1 < \lambda_i(E) < 1, i = 0, 1, \cdots, n. \tag{31}$$

Furthermore, we have the following implication:

$$0 < \rho(E) < 1, \tag{32}$$

$\rho(E)$ is the spectral radius of E. Therefore, we can derive the following:

$$\lim_{k \to \inf} E^k = \mathbf{0}, \tag{33}$$

Moreover, it is evident that as k approaches infinity, m tends to 1, and M approaches the identity matrix I. According to equation (29). we can know

$$\begin{aligned} &\lim_{k \to \inf} \mathbf{P}^{k+1} \\ &= (M^k - M^k \mu A^T A - I)^{-1}(M^k \mu A^T Q) \\ &+ (1 - \mu A^T A)^{k+1}[\mathbf{P}^0 - (M^0 - M^0 \mu A^T A - I)^{-1}(M^0 \mu A^T Q)] \\ &= A^{-1} Q \end{aligned} \tag{34}$$

Hence, it is proven that GLS-PIA converges. □

6. Experiments

To validate the superiority and generality of our method, we conducted several comparative experiments. Firstly, to verify the effectiveness of the adaptive knot placement method, we compared the fitting accuracies obtained using different knot placement methods. Secondly, we compared our fitting framework with other methods regarding fitting accuracy. For the sake of visualization convenience, the examples from S^2 are selected as the subjects for the first two experiments. Finally, we applied our method to three scenarios, including fitting the Earth's national borders, rigid body rotation, and shape space animation, demonstrating the generality of our method on spherical surfaces of different dimensions.

We used two metrics to measure the accuracy of the fitting methods: maximum error and mean error.

$$Max\_Error = \max_i cos^{-1}[p_i \cdot C(u_i)] \tag{35}$$

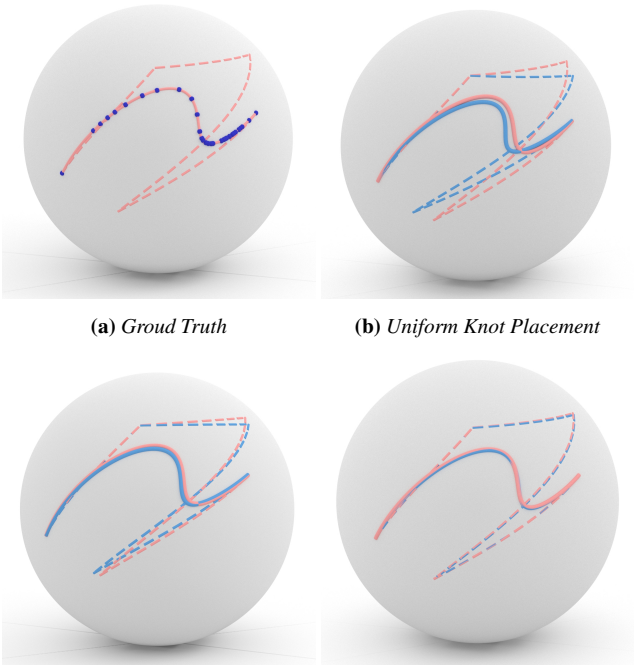
$$Mean\_Error = \frac{1}{m} \sum_{i=1}^{m} cos^{-1}[p_i \cdot C(u_i)] \tag{36}$$

All experiments are implemented on MATLAB with an Intel Core i7-10700K CPU @ 3.80 GHz and 16 GB of memory. The blue dots represent the data points, the blue lines depict the results from different methods, and the pink line represents the ground truth. The solid lines represent the spline curves, while the dashed lines represent the corresponding control polygons.

In this subsection, we designed two experimental examples to demonstrate the excellent performance of our adaptive knot placement method, which we refer to as the simple example (see Figgure 7) the complex example (see Figure 8). We conducted experiments on these two examples using three different knot placement methods: uniform knot placement, euclidean difference estimate knot placement based on R^{n+1} space, and geodesic difference estimate knot placement based on S^n (ours).

Both examples are sampled from a spherical B-spline curve. The first example is a simple example in which the distribution of fitting points is relatively uniform, and the sampling is also relatively uniform. According to our expectations, the three knot placement methods will perform relatively similarly on this example. The second example is a complex example in which there is obvious imbalance in detail information and sampling. According to our expectations, the three methods will exhibit significant differences in fitting accuracy on this example.

6.1. Comparision on different Knot Placement



(a) *Groud Truth* **(b)** *Uniform Knot Placement*

(c) *Euclidean Difference* **(d)** *Geodesic Difference*

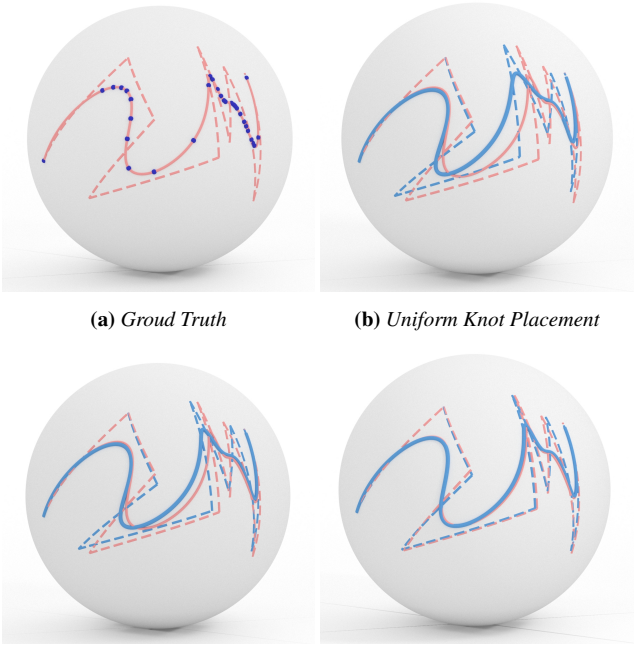
Figure 7: *Fitting result of the simple example by different knot placement method.*

To highlight these performance differences, we keep the number of iterations for each fitting process at 30 to emphasize the impact of different knot placement methods on fitting accuracy.

As shown in the Figure 7, on the simple example, the performance of the three methods is relatively similar, all demonstrating good fitting performance. Only the uniform knot placement method had lower fitting accuracy, mainly due to poor capture of detail in-

Table 1: *Numerical comparison of different knot placement method on simple example in Figure 7.*

Metrics	Uniform Knot placement	Euclidean Difference	Geodesic Difference
Max_Error	1.736×10^{-1}	1.02×10^{-2}	$\mathbf{3.9\times10^{-3}}$
Mean_Error	8.34×10^{-2}	6.4×10^{-3}	$\mathbf{1.2\times10^{-3}}$



(a) *Groud Truth* **(b)** *Uniform Knot Placement*

(c) *Euclidean Difference* **(d)** *Geodesic Difference*

Figure 8: *Fitting result of the complex example by different knot placement method.*

Table 2: *Numerical comparison of different knot placement method on complex example in Figure 8.*

Metrics	Uniform Knot placement	Euclidean Difference	Geodesic Difference
Max_Error	5.239×10^{-1}	2.032×10^{-1}	$\mathbf{1.08\times10^{-2}}$
Mean_Error	2.983×10^{-1}	1.427×10^{-1}	$\mathbf{6.2\times10^{-3}}$

formation in the bent region. The table 1 shows the numerical comparison of the fitting accuracy of the three methods.

On the complex example, as shown in the Figure 8, there is a significant difference in performance among the three methods, with uniform knot placement being the worst, Euclidean difference estimate knot placement based being relatively good, and our method being the best. We found that the main difference among the three methods is not entirely in the complex detail area on the right side of the curve but in the transitional part in the middle of the curve. This is because there are more sampling points on the right side of the curve, resulting in larger adjustment vectors during fitting, so the control points on the right side are updated more quickly, leading to better fitting performance in the complex detail area on the right side. However, for the first two knot placement methods, due to the initial distribution of control points being less reasonable than our method, more control points appear in the left and middle parts with fewer details, and smaller adjustment vectors due to the relatively few sampling points are allocated to more control points, resulting in slower fitting and lower accuracy in that part. The table 2 shows the numerical comparison of the fitting accuracy of the three methods.

6.2. Comparision on different Fitting Method

In our experiments, we compared our method with four mainstream methods: **Gousenbourger's Composite Bézier-Like Curves** [GMA18], **Gousenbourger's Blended Cubic Splines** [GMA18], **Samir's Gradient-Descent method** [SASK12], and **Jupp's Unwrapping method** [JK18].

We select the results of our method after 30 iterations in the experiments. For Samir's Gradient-Descent method and Jupp's Unwrapping method, we run their iterative optimization until the same amount of time as our 30 iterations have passed. We observe that these two methods had almost converged within this iteration time. Gousenbourger's composite Bézier-like curves and blended cubic splines do not require a threshold or stopping condition to terminate optimization, and their optimization process is finite, so we did not impose time limits on them.

We used the maximum error and mean error metrics to evaluate the fitting accuracy of each method. The results showed that our method outperformed the other methods in terms of fitting accuracy, and the improvement is significant. The table 3 shows the numerical comparison of the fitting accuracy of the five methods. Compared to the best existing method on the this example, our method reduces the Max_error by 51% and the mean_error by 37%. Figure 9 shows visual results of the five methods.

Furthermore, we present several additional examples to showcase the distinctive characteristics of our approach. In the case of the helix curve example 10, the distribution of detailed features within the curve itself is relatively uniform, and the provided data points exhibit a correspondingly even distribution. As a result, different methods achieve commendable fitting results.

In the case of the Shock line example 11, the distribution of data points is no longer uniform, giving rise to an imbalanced scenario. Other methods exhibit significant fitting errors in regions with fewer data points, whereas our method successfully preserves the curve's characteristics even in such data-sparse regions.

When dealing with curves containing sharp corners 12, our method outperforms others in faithfully capturing the sharp corner features. This capability stems from our adaptive knot placement method, which adeptly captures local detailed features. However, due to the inherent nature of B-spline curves, accurately reproducing sharp inflection points remains challenging and necessitates approximations using high-curvature curves for precise fitting.

6.3. Applications

In this subsection, we apply our method to three tasks: fitting national borders on the Earth's sphere, shape interpolation based on Kendall shape space, and three-dimensional rigid body rotation.

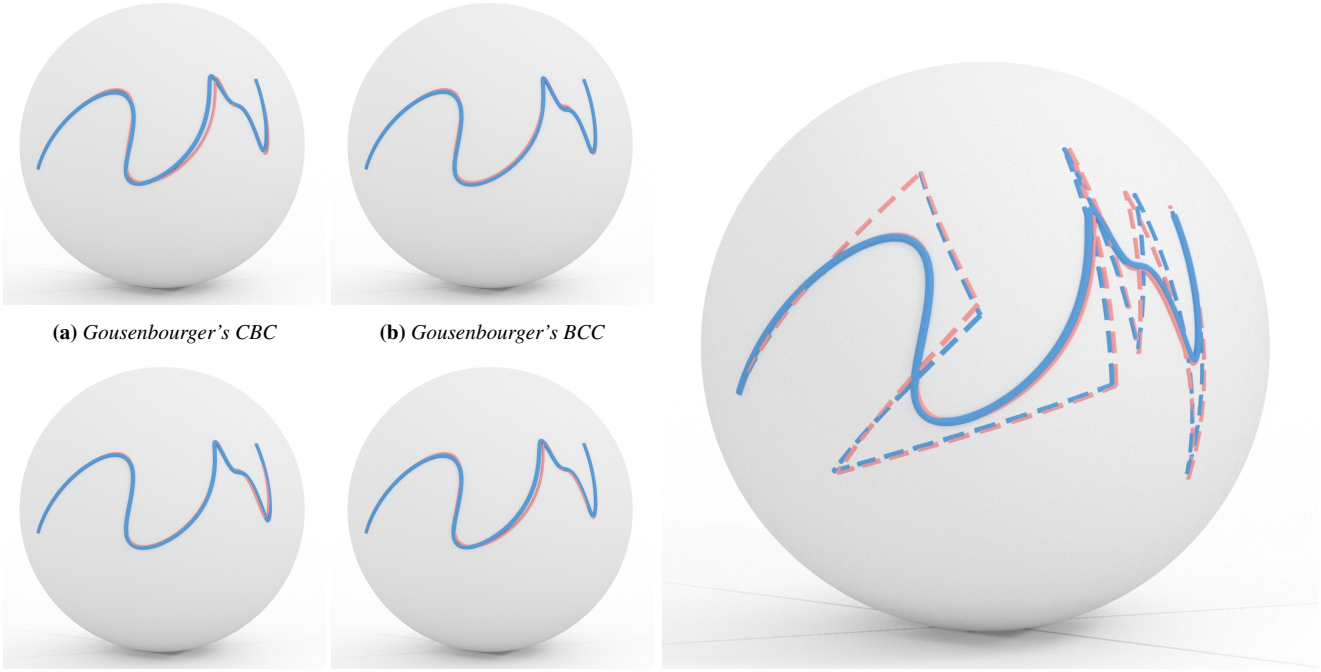
For the task of fitting national borders on the Earth's sphere, we selected a segment of the US-Mexico border as an example and chose a series of discrete points on it as input for fitting. We have selected two sets of data with different resolutions, as shown in Figure 13. We performed fitting on the low-resolution and high-resolution data and compared the results with them. Despite the lack of fine details resulting from the information loss in the low-resolution data, our method still demonstrates good fitting performance. When we utilize high-resolution data for fitting, we achieve excellent results. As shown in the Figure 13, our method produced good results in restoring the details of the curve.

Kendall shape space is a discrete shape space that can express shape attributes through landmark points while eliminating translation, rotation and scaling. Kendall shape space is essentially a high-dimensional sphere, and each shape can be expressed as a point on Kendall shape space. A curve on Kendall shape space can express a shape transformation animation. As shown in the Figure 14, we collect landmark points for the black character shape and map it to several points on Kendall shape space, then fit a curve to obtain a continuous transformation of the shape. We sample several intermediate shapes (shown in red) from it for display.

Three-dimensional rigid body rotation can be expressed as a quaternion, and the rotation quaternion space is essentially an S^3 sphere. We expressed several rotation states of the 3D model as points on S^3 sphere and fit them to obtain a continuous representation of the rotation of the 3D model. We sample some points from the continuous rotation transformation and visualized them as shown in the Figure 15.

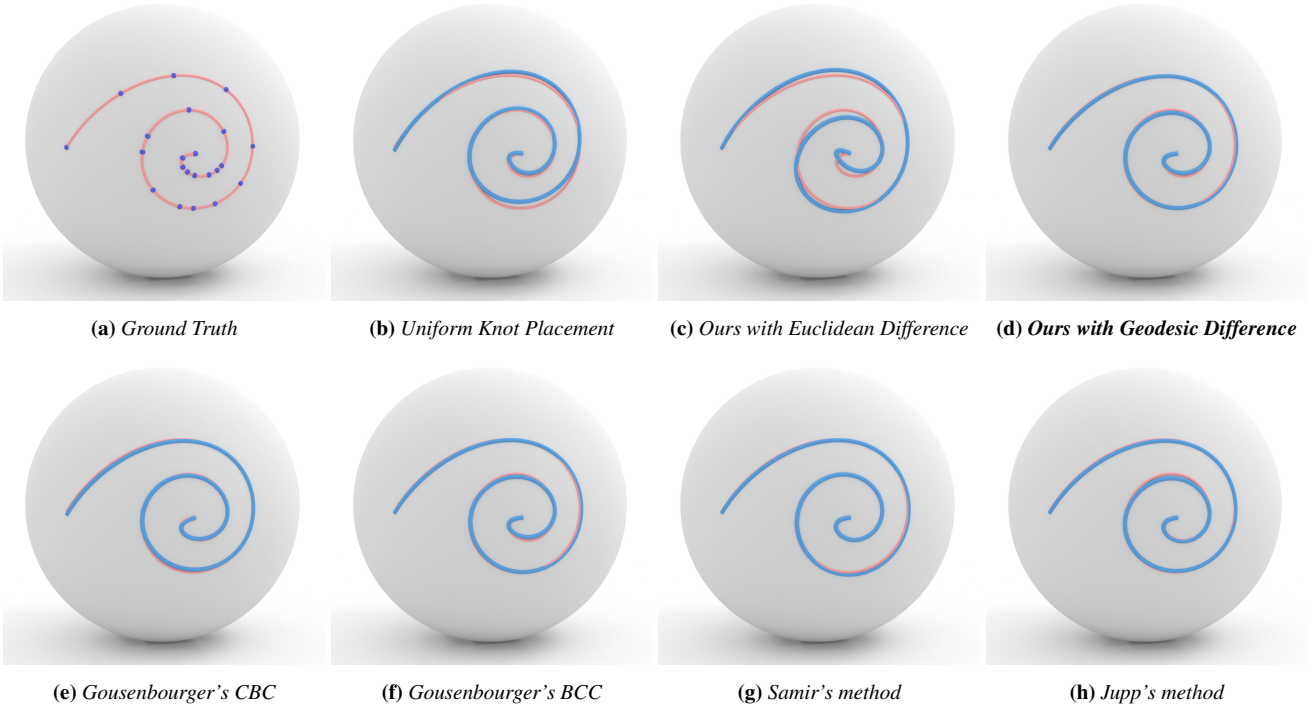
7. Conclusions

In this paper, we proposed a spherical B-spline curve fitting framework based on geodesic least squares and adaptive knot placement. The adaptive knot placement based on geodesic difference enables B-spline curves to better capture the distribution of details in the input data points on spheres, thereby improving the fitting accuracy with less control points. We used geodesic least squares loss as the optimization objective and proposed the GLS-PIA method to overcome the difficulty of optimizing least squares loss on the sphere. Meanwhile, we used an error allocation method based on control point contribution coefficient to limit the number of control points



(a) *Gousenbourger's CBC* **(b)** *Gousenbourger's BCC* **(c)** *Samir's method* **(d)** *Jupp's method* **(e)** *Ours*

Figure 9: *Fitting result by different fitting method.*

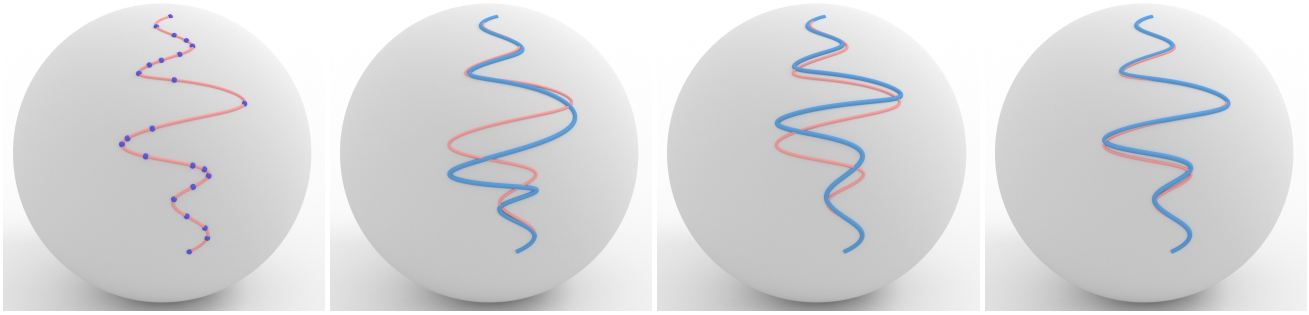


(a) *Ground Truth* **(b)** *Uniform Knot Placement* **(c)** *Ours with Euclidean Difference* **(d)** ***Ours with Geodesic Difference*** **(e)** *Gousenbourger's CBC* **(f)** *Gousenbourger's BCC* **(g)** *Samir's method* **(h)** *Jupp's method*

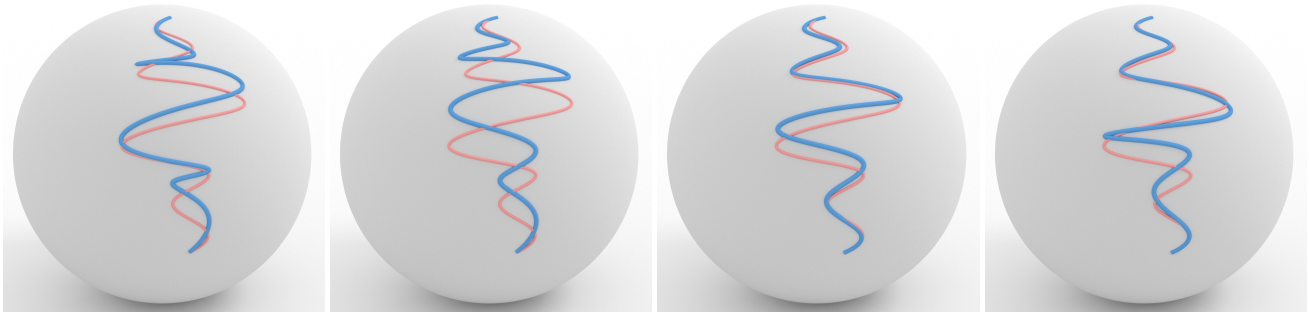
Figure 10: *Fitting result by different fitting methods on Helix curve.In the case of the helix curve example, the distribution of detailed features within the curve itself is relatively uniform, and the provided data points exhibit a correspondingly even distribution. As a result, different methods achieve commendable fitting results.*

Table 3: *Numerical comparison of different fitting method in Figure 9.*

Metrics	Gousenbourger's CBC	Gousenbourger's BCC	Samir's method	Jupp's method	Ours.
Max_Error	1.225×10^{-1}	8.32×10^{-2}	1.194×10^{-1}	2.19×10^{-2}	$\mathbf{1.08\times10^{-2}}$
Mean_Error	6.92×10^{-2}	3.27×10^{-2}	5.37×10^{-2}	9.8×10^{-3}	$\mathbf{6.2\times10^{-3}}$



(a) *Ground Truth* **(b)** *Uniform Knot Placement* **(c)** *Ours with Euclidean Difference* **(d)** ***Ours with Geodesic Difference***



(e) *Gousenbourger's CBC* **(f)** *Gousenbourger's BCC* **(g)** *Samir's method* **(h)** *Jupp's method*

Figure 11: *Fitting result by different fitting methods on Shock line. In the case of the Shock line example, the distribution of data points is no longer uniform, giving rise to an imbalanced scenario. Other methods exhibit significant fitting errors in regions with fewer data points, whereas our method successfully preserves the curve's characteristics even in such data-sparse regions.*

in the PIA method to ensure the fairness of the fitted curve. With our framework, it is possible to achieve both accuracy and fairness in spherical B-spline curve fitting. Experimental results demonstrate that our method has good performance and generality.

Currently, the method presented in this paper can only be applied to sphere manifolds and relies on the orderedness of the input data points. In the future, we will focus on spline curve fitting problems and unordered point fitting problems on general manifolds, as well as explore and expand more related applications.

Acknowledgement

The authors want to thank the anonymous reviewers for their constructive comments. This research was partially supported by the National Nature Science Foundation of China(No.62072045), the Beijing Municipal Science and Technology Commission and Zhongguancun Science Park Management Committee (No.Z221100002722020), and the Innovation Transfer Fund of Peking University Third Hospital(No.BYSYZHKC2021110).

References

[AEC*18] AGUILAR E., ELIZALDE H., CÁRDENAS D., PROBST O., MARZOCCA P., RAMIREZ-MENDOZA R.: An adaptive curvature-guided approach for the knot-placement problem in fitted splines sci-journal. *Journal of Computing and Information Science in Engineering 18* (07 2018). 4

[AMAS16] ALDERSON T., MAHDAVI-AMIRI A., SAMAVATI F.: Multiresolution on spherical curves. *Graphical Models 86* (2016), 13–24. 1, 3

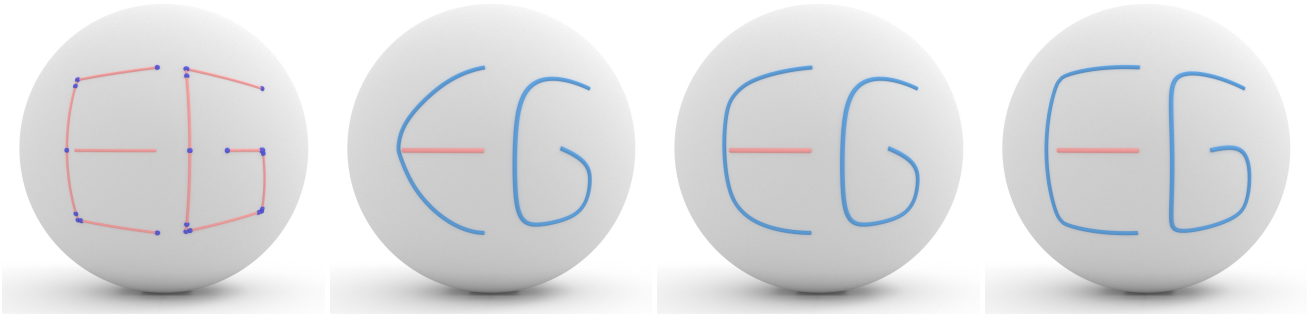
[AS19] ALDERSON T., SAMAVATI F.: Multiscale nurbs curves on the sphere and ellipsoid. *Computers & Graphics 82* (2019), 243–249. 1, 3

[CKL99] CROUCH P., KUN G., LEITE F. S.: The de casteljau algorithm on lie groups and spheres. *Journal of Dynamical and Control Systems 5*, 3 (1999), 397–429. 3

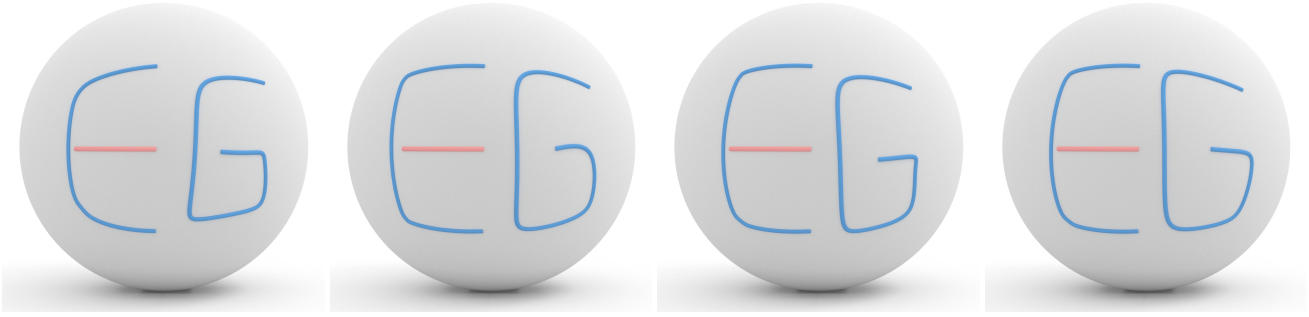
[CLCQ12] CAO J., LI X., CHEN Z., QIN H.: Spherical dcb-spline surfaces with hierarchical and adaptive knot insertion. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1290–1303. 3

[CMRS01] CONTI C., MORANDI R., RABUT C., SESTINI A.: Cubic spline data reduction choosing the knots from a third derivative criterion. *Numerical Algorithms 28*, 1-4 (2001), 45–61. 4

[GMA18] GOUSENBOURGER P. Y., MASSART E., ABSIL P. A.: Data



(a) *Ground Truth* **(b)** *Uniform Knot Placement* **(c)** *Ours with Euclidean Difference* **(d)** ***Ours with Geodesic Difference***



(e) *Gousenbourger's CBC* **(f)** *Gousenbourger's BCC* **(g)** *Samir's method* **(h)** *Jupp's method*

Figure 12: *Fitting result by different fitting method on curve with sharp corner. When dealing with curves containing sharp corners, our method outperforms others in faithfully capturing the sharp corner features. This capability stems from our adaptive knot placement method, which adeptly captures local detailed features.*

fitting on manifolds with composite bézier-like curves and blended cubic splines. *Journal of Mathematical Imaging Vision* (2018). 2, 3, 10

[HGFL15] HUANG K.-T., GONG H., FANG F., LI Z.: Generation of spherical non-uniform rational basis spline curves and its application in five-axis machining. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 229* (07 2015). 1, 2, 3

[HLGQ05] HE Y., LI X., GU X., QIN H.: Brain image analysis using spherical splines. In *Energy Minimization Methods in Computer Vision and Pattern Recognition* (Berlin, Heidelberg, 2005), Rangarajan A., Vemuri B., Yuille A. L., (Eds.), Springer Berlin Heidelberg, pp. 633–644. 1

[JK18] JUPP P. E., KENT J. T.: Fitting Smooth Paths to Spherical Data. *Journal of the Royal Statistical Society Series C: Applied Statistics 36*, 1 (12 2018), 34–46. 2, 3, 10

[KDLS20] KIM K.-R., DRYDEN I. L., LE H., SEVERN K. E.: Smoothing Splines on Riemannian Manifolds, with Applications to 3D Shape Space. *Journal of the Royal Statistical Society Series B: Statistical Methodology 83*, 1 (12 2020), 108–132. 1, 2

[KKL10] KIM KENOBI I. L. D., LE H.: Shape curves and geodesic modelling. *Biometrika 97*, 3 (2010), 567–584. 3

[KKS95a] KIM M.-J., KIM M.-S., SHIN S.: General construction scheme for unit quaternion curves with simple high order derivatives. pp. 369–376. 3

[KKS95b] KIM M.-J., KIM M.-S., SHIN S. Y.: A c/sup 2/-continuous b-spline quaternion curve interpolating a given sequence of solid orientations. In *Proceedings Computer Animation'95* (1995), pp. 72–81. 3

[KKS96] KIM M.-J., KIM M.-S., SHIN S. Y.: A compact differential formula for the first derivative of a unit quaternion curve. *The Journal of Visualization and Computer Animation 7*, 1 (1996), 43–57. 3

[KN95] KIM M.-S., NAM K.-W.: Interpolating solid orientations with circular blending quaternion curves. *Computer-Aided Design 27*, 5 (1995), 385–398. 3

[LZJ*17] LIANG F., ZHAO J., JI S., FAN C., ZHANG B.: A novel knot selection method for the error-bounded b-spline curve fitting of sampling points in the measuring process. *Measurement Science Technology 28*, 6 (1 2017), 065015–1–065015–14. 4

[MKM*99] MORRIS R. J., KENT J. T., MARDIA K. V., FIDRICH M., LINNEY A.: Analysing growth in faces. In *Int Conf. on Imaging Science Systems and Technology (CISST '99), in (H. R. Arabnia Ed.)* (1999). 3

[PD79] PARKER R. L., DENHAM C. R.: Interpolation of unit vectors. *Geophysical Journal International 58*, 3 (09 1979), 685–687. 3

[PN06] POPIEL T., NOAKES L.: C2 spherical bézier splines. *Computer Aided Geometric Design 23*, 3 (2006), 261–275. 3

[Pre87] PRENTICE M. J.: Fitting smooth paths to rotation data. *Journal of the Royal Statistical Society: Series C (Applied Statistics) 36*, 3 (1987), 325–331. 3

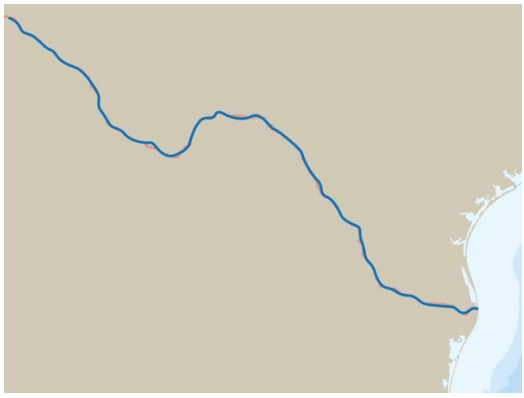
[SASK12] SAMIR C., ABSIL P. A., SRIVASTAVA A., KLASSEN E.: A gradient-descent method for curve fitting on riemannian manifolds. *Foundations of Computational Mathematics 12*, 1 (2012), 49–73. 2, 3, 10

[SDK*12] SU J., DRYDEN I., KLASSEN E., LE H., SRIVASTAVA A.: Fitting smoothing splines to time-indexed, noisy points on nonlinear manifolds. *Image and Vision Computing 30*, 6 (2012), 428–442. 1, 2

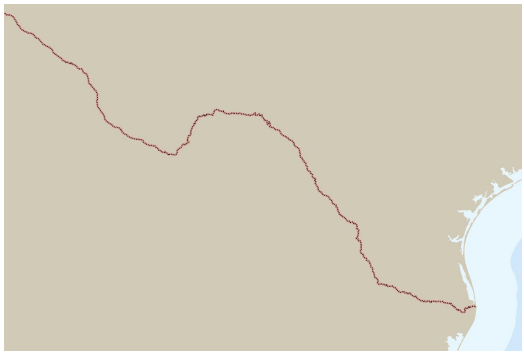
[Sho85] SHOEMAKE K.: Animating rotation with quaternion curves. *Computer Graphics* (1985). 1, 2, 3

[SHSH07] SHIMOSAKA A., HAYASHI K., SHIRAKAWA Y., HIDAKA J.: *On the Geometry of Rolling and Interpolation Curves on Sn, SOn, and*

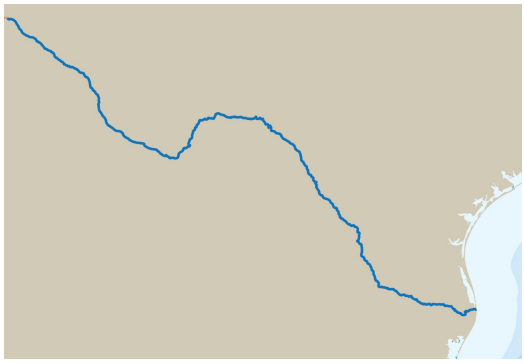
(a) *Input points with the low-resolution data.*



(b) *Fitting result with the low-resolution data..*



(c) *Input points with the high-resolution data.*



(d) *Fitting result with the high-resolution data.*

Figure 13: *US-Mexico border line fitting.*

Grassmann Manifolds. On the Geometry of Rolling and Interpolation Curves on Sn, SOn, and Grassmann Manifolds, 2007. 3

[TDH15] TJAHJOWIDODO T., DUNG V., HAN M.: A fast non-uniform knots placement method for b-spline fitting. In *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)* (2015), pp. 1490–1495. 4

[TXFH18] TAN J., XING Y., FAN W., HONG P.: Smooth orientation interpolation using parametric quintic-polynomial-based quaternion spline curve. *North-Holland* (2018). 3

[Uk07] UK A. K. A. K. A.: Shape-space smoothing splines for planar landmark data. *Biometrika 94*, 3 (2007), 513–528. 3

[XQ01] XIE H., QIN H.: Automatic knot determination of nurbs for interactive geometric design. In *Proceedings International Conference on Shape Modeling and Applications* (2001), pp. 267–276. 4

[YNPT20] YEH R., NASHED Y. S., PETERKA T., TRICOCHE X.: Fast automatic knot placement method for accurate b-spline curve fitting. *Computer-Aided Design 128* (2020), 102905. 4

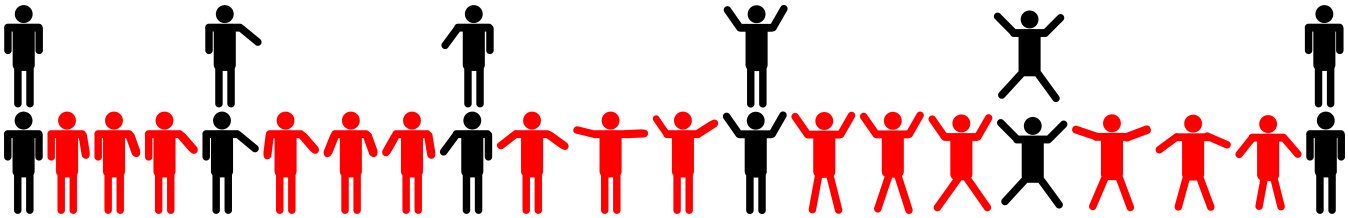


Figure 14: *Kendall shape space example. The black characters correspond to the data points to be fitted on the Kendall sphere, and the red characters correspond to the newly generated points on the fitting curve.*

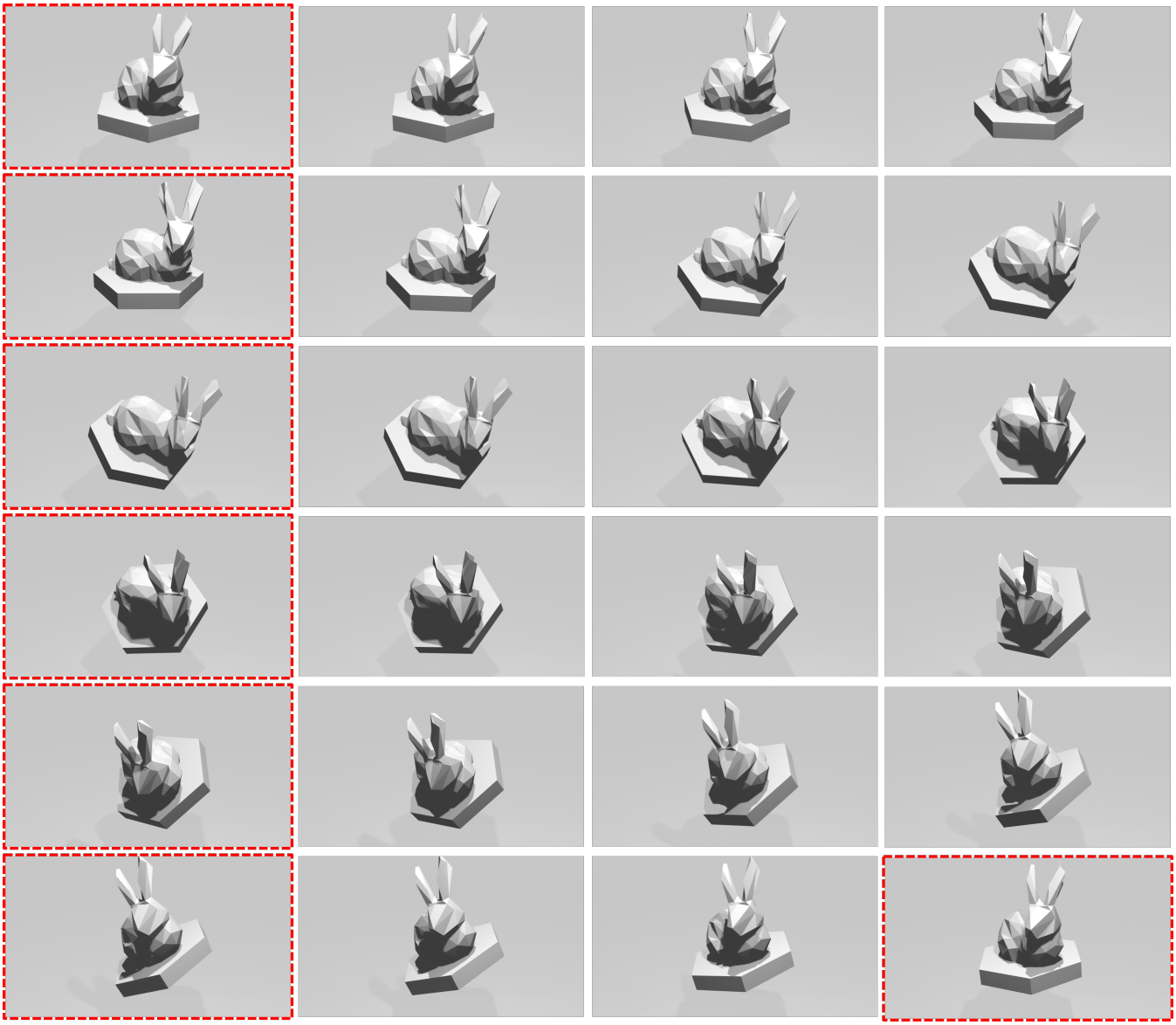


Figure 15: *Bunny roation. We can view each rotation state as a point on S^3 sphere. The red dashed box corresponds to the fitting data points, while the rest correspond to the newly generated points on the fitting curve.*