Li X, Wu Z, Pan F *et al.* A geometric strategy algorithm for orthogonal projection onto a parametric surface. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 34(6): 1279–1293 Nov. 2019. DOI 10.1007/s11390-019-1967-z

A Geometric Strategy Algorithm for Orthogonal Projection onto a Parametric Surface

Xiaowu Li[1], Zhinan Wu[2], Feng Pan[3], *Senior Member*, *CCF*, Juan Liang[4], Jiafeng Zhang[1] and Linke Hou[5,*]

[1] *College of Data Science and Information Engineering, Guizhou Minzu University, Guiyang 550025, China*

[2] *School of Mathematics and Computer Science, Yichun University, Yichun 336000, China*

[3] *School of Software Engineering, South China University of Technology, Guangzhou 510006, China*

[4] *Department of Science, Taiyuan Institute of Technology, Taiyuan 030008, China*

[5] *Center for Economic Research, Shandong University, Jinan 250100, China*

E-mail: lixiaowu002@126.com; zhi_nan_7@163.com; panf@vip.163.com; liangjuan76@126.com
{jiafengzhang, abram75}@163.com

Received January 13, 2019; revised September 4, 2019.

Abstract In this paper, we investigate how to compute the minimum distance between a point and a parametric surface, and then to return the nearest point (foot point) on the surface as well as its corresponding parameter, which is also called the point projection problem of a parametric surface. The geometric strategy algorithm (hereafter GSA) presented consists of two parts as follows. The normal curvature to a given parametric surface is used to find the corresponding foot point firstly, and then the Taylor's expansion of the parametric surface is employed to compute parameter increments and to get the iteration formula to calculate the orthogonal projection point of test point to the parametric surface. Our geometric strategy algorithm is essentially dependent on the geometric property of the normal curvature, and performs better than existing methods in two ways. Firstly, GSA converges faster than existing methods, such as the method to turn the problem into a root-finding of nonlinear system, subdividing methods, clipping methods, geometric methods (tangent vector and geometric curvature) and hybrid second-order method, etc. Specially, it converges faster than the classical Newton's iterative method. Secondly, GSA is independent of the initial iterative value, which we prove in Theorem 1. Many numerical examples confirm GSA's robustness and efficiency.

Keywords point projection problem, point inversion problem, normal curvature, normal curvature sphere, convergence analysis

1 Introduction

In this paper, we discuss how to compute the minimum distance between a point and a parametric surface, and to return the nearest point (footpoint) on the surface as well as its corresponding parameter, which is also called the point projection problem (the point inversion problem) of a parametric surface. It is a very interesting problem in geometric modeling, computer graphics and computer vision[1]. Both projection and inversion are essential for interactively selecting curves and surfaces[1−2], for the curve fitting problem[1−2], for reconstructing surfaces[3−5] and for projecting of a space curve onto a surface in surface curve design[1]. It is also a key issue in the ICP (iterative closest point) algorithm for shape registration[6].

The most classical method for solving nonlinear equation or system of nonlinear equations is Newton-

Regular Paper

This work is supported by the National Natural Science Foundation of China under Grant No. 61263034, the Feature Key Laboratory for Regular Institutions of Higher Education of Guizhou Province of China under Grant No. [2016]003, the Key Laboratory of Advanced Manufacturing Technology of Ministry of Education of China with Guizhou University under Grant No. KY[2018]479, the Training Center for Network Security and Big Data Application of Guizhou Minzu University under Grant No. 20161113006, the Shandong Provincial Natural Science Foundation of China under Grant No. ZR2016GM24, the Progress Project for Young Science and Technology Scholars of Guizhou Provincial Department of Education under Grant No. KY[2016]164.

*Corresponding Author

Raphson method[7−11]. Mortenson[7] turned the projection problem into the one to find the root of a polynomial by using the Newton-Raphson method. Zhou *et al.*[8] presented an algorithm to find the stationary points of the squared distance functions between two point sets by solving equations expressed in the tensor product Bernstein basis. Limaien and Trochu[9] computed the orthogonal projection of a point onto parametric curves and surfaces by constructing an auxiliary function and finding its zeros. Polak and Royset[10] presented a new feedback precision-adjustment rule for use with a smoothing technique and standard unconstrained minimization algorithms in the solution of finite minimax problems. Patrikalakis and Maekawa[11] transformed distance functions problem to solving systems of nonlinear polynomial equations. The common technique is to turn the problem into the one to use the Newton's iterative method for finding the roots of a nonlinear equation system, which is dependent on the initial iterative value.

The second classical method for orthogonally projecting a point onto parametric curve or surface is the subdividing method[1,12−15]. By subdividing Bézier curve or surface and making use of the relationship between control points and curve or surface (the control points being a very good sample of the target, including derivatives), Ma and Hewitt[1] proposed an algorithm to project a point onto a parametric curve or surface. Johnson and Cohen[12] presented a robust search for distance extrema between a point and a curve or a surface for finding all local extrema. Based on the algorithm of Ma and Hewitt[1], Selimovic[13] presented improved algorithms for the projection of points onto NURBS curve and surface. Cohen *et al.*[14] provided classical subdivision algorithms, which have been widespreadly applied in computer-aided geometric design, computer graphics, and numerical analysis. Piegl and Tiller[15] presented an algorithm for point projection onto the NURBS surface by subdividing a NURBS surface into quadrilaterals, projecting the test point onto the closest quadrilateral, and then recovering the parameter from the closest quadrilateral. The common feature in this branch of the literature is to use the subdivision method firstly, and to use the Newton's iterative method in the last step. The method with subdivision processing is time-consuming, while the Newton's iterative method used in the last step to find the roots for a nonlinear equation system depends on the initial iterative value.

The third classical method transforms the point projection problem into specific solvers' methods[16−20]. By using multivariate rational functions, Elber and Kim[16] established a solver for a set of geometric constraints represented by inequalities. When the dimension of the solver is larger than zero, they subdivided the multivariate function(s) in order to designate the function values to a specified domain. Borrowing from [16] but with more effectiveness, a hybrid parallel method in [17] develops both the CPU and the GPU multi-core architectures to figure out systems under multivariate constraints. Those GPU-based subdivision methods essentially explore the inherent parallelism in the subdivision of multivariate polynomial. Compared with the existing subdivision-based CPU, the performance of the geometric-based algorithm has been improved to a certain extent. Two blending schemes in [18] efficiently eliminate no-root domains, and thereby greatly decrease the number of subdivisions. For a nonlinear equation system, a simple linear combination of functions can remove no-root domain and then find out all control points for its Bernstein-Bézier bases with the same sign, which must be consistent with the seek function. During the subdivision process, it can continuously yield these types of functions to eliminate the no-root domain. As a consequence, van Sosin and Elber[19] efficiently constructed a variety of complex piecewise polynomial systems with zero or inequality constraints in zero-dimensional or one-dimensional solution spaces. On the basis of their own studies[16,19], Bartoň *et al.*[20] came up with a new solver to solve a non-constrained (piecewise) polynomial system. Two termination criteria are adopted in the subdivision-based solver: the no-loop test and the single-component test. Once the subdivision-based solver has met the two termination criteria, it then can obtain the domains which have a single monotone univariate solution. The advantage of these methods is that they can find all the root solutions, while their disadvantage is that they are computationally expensive and may need many subdivision steps.

The fourth classical method for the point projection problem is clipping technique[21−23]. Chen *et al.*[21,22] provided methods for computing the minimum distance between a point and a NURBS curve (or a clamped B-spline surface). Analogously, based on an efficient culling technique to eliminate redundant curves or surfaces with no projection from the given point, Oh *et al.*[23] presented an efficient algorithm for projecting a given point to its closest point on a family of freeform curves and surfaces. This branch of the literature uses

the clipping methods[21−23] and then uses the Newton's iterative method in the last step. But the method with clipping processing is time-consuming and the Newton's iterative method in the last step depends on the initial iterative value.

The fifth classical method for the point projection problem makes use of five types of geometric methods: tangent cones method[12], torus patch method[24], curvature information method[25−27], tangent vector method[28−30] and geometric hybrid method[31−33]. Using geometric operations with tangent cones rather than numerical methods, Johnson and Cohen[12] presented a robust search for distance extrema between a point and a curve or a surface for finding all local extrema. The technique of torus patch approximation to a local surface was proposed by Liu *et al.*[24] It was proved that the approximation torus patch and the original surface are both second-order osculating. An algorithm with curvature information for orthogonal projection onto curves and surfaces has been presented by Hu and Wallner[25]. Li *et al.*[26] presented a second-order curvature geometry method for computing the minimum distance between a point and a spatial parametric curve. Li *et al.*[27] gave a convergence analysis of the point projecting onto the planar parametric curve algorithm in [25]. Utilizing the tangent vector method, the first-order algorithm for point projection problem (point inversion problem) of a parametric curve or surface has been realized by Hartmann[28], Hoschek and Lasser[29] and Hu *et al.*[30], respectively (hereafter H-H-H method). For some special cases where the H-H-H method diverges, Liang *et al.*[31] created the hybrid second-order method for orthogonal projection onto parametric curve in an n-dimensional Euclidean space. Besides, Li *et al.*[32] presented the hybrid second-order iterative algorithm for orthogonal projection onto parametric surface. The robustness of the two algorithms[31,32] is improved compared with the H-H-H method. Based on the tangent vector method and combining with Newton's iterative method, Li *et al.*[33] presented an integrated hybrid second-order algorithm for orthogonal projection onto a planar implicit curve. This branch of literature shares the adoption of geometric methods such as tangent cone, tangent vector, torus patch, curvature information or geometric hybrid. But their geometry convergence rates are not very fast.

In a word, these algorithms have been presented to investigate various techniques such as Newton's iterative method, root-finding methods, subdividing methods, clipping technique, transform-based solvers methods and various geometric methods. Regarding the projection problem, Ko and Sakkalis[34] systematically and completely summarized literatures before 2014. To avoid the algorithms' dependence on the initial iterative value, we use the geometric iterative strategy. It uses only such second-order information of the surface under consideration. We firstly construct the normal curvature sphere to a given parametric surface, and then specify its radius and center. Secondly, we find the footpoint on the line segment between the test point and the center of the normal curvature sphere. Finally we use the Taylor's expansion of the surface to compute parameter increments and get the iteration formula to compute the orthogonal projection point of the test point to the parametric surface (see Fig.1). We prove that GSA is independent of the initial iterative value. Numerical examples are shown to illustrate the efficiency and robustness of the geometric iterative strategy.

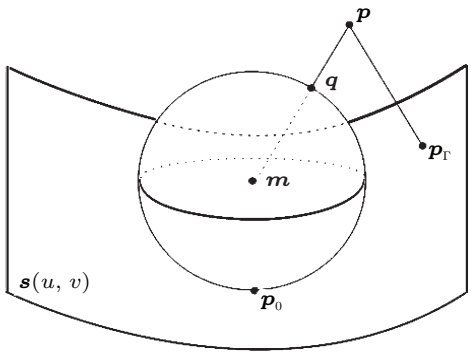


Fig.1. Graphic demonstration for GSA.

The rest of this paper is organized as follows. Section 2 presents GSA for the point projection problem and Section 3 describes its convergence analysis. The experimental results about the evaluation of performance of various methods are given in Section 4. Finally, Section 5 concludes the paper.

2 Orthogonal Projection onto a Parametric Surface

2.1 Description of GSA

Assume that a surface has a parametric form Γ: $\boldsymbol{s}(u,v) = (f_1(u,v), f_2(u,v), f_3(u,v))$ in $\mathbb{R}^3$. The scalar product of vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3$ is $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Partial derivatives with respect to the parameters u and v will be denoted by $\frac{\partial \boldsymbol{s}}{\partial u} = (\frac{\partial f_1(u,v)}{\partial u}, \frac{\partial f_2(u,v)}{\partial u}, \frac{\partial f_3(u,v)}{\partial u})$ and $\frac{\partial \boldsymbol{s}}{\partial v} =$

$(\frac{\partial f_1(u,v)}{\partial v}, \frac{\partial f_2(u,v)}{\partial v}, \frac{\partial f_3(u,v)}{\partial v})$. The unit normal vector of the parametric surface $\boldsymbol{s}(u,v)$ on the point $\boldsymbol{s}(u_0,v_0)$ could be defined as $\boldsymbol{n} = \frac{\frac{\partial \boldsymbol{s}}{\partial u}\times\frac{\partial \boldsymbol{s}}{\partial v}}{\|\frac{\partial \boldsymbol{s}}{\partial u}\times\frac{\partial \boldsymbol{s}}{\partial v}\|}|_{(u_0,v_0)}$. A test point $\boldsymbol{p}$ ($\boldsymbol{p} = (p_1,p_2,p_3)$) is projected onto a surface as follows. Assume an initial iteration point $\boldsymbol{p}_0 = \boldsymbol{s}(u_0,v_0)$, and then we find $\boldsymbol{q}$ by projecting test point $\boldsymbol{p}$ onto the tangent plane determined by $\boldsymbol{p}_0$.

$$\boldsymbol{q} - \boldsymbol{p}_0 \approx \frac{\partial \boldsymbol{s}}{\partial u}\Delta u + \frac{\partial \boldsymbol{s}}{\partial v}\Delta v. \tag{1}$$

Multiplying both sides of (1) by $\frac{\partial \boldsymbol{s}}{\partial u}$, $\frac{\partial \boldsymbol{s}}{\partial v}$, respectively, we get

$$\begin{cases} \langle\frac{\partial \boldsymbol{s}}{\partial u}, \frac{\partial \boldsymbol{s}}{\partial u}\rangle\Delta u + \langle\frac{\partial \boldsymbol{s}}{\partial u}, \frac{\partial \boldsymbol{s}}{\partial v}\rangle\Delta v = \langle \boldsymbol{q}-\boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial u}\rangle, \\ \langle\frac{\partial \boldsymbol{s}}{\partial u}, \frac{\partial \boldsymbol{s}}{\partial v}\rangle\Delta u + \langle\frac{\partial \boldsymbol{s}}{\partial v}, \frac{\partial \boldsymbol{s}}{\partial v}\rangle\Delta v = \langle \boldsymbol{q}-\boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial v}\rangle, \end{cases} \tag{2}$$

where symbol $\langle\ ,\ \rangle$ is the inner product. Therefore $\Delta u, \Delta v$ can be computed as a solution of a regular system of linear equations in (2). We update u_0, v_0 by adding $\Delta u, \Delta v$, respectively. This first-order geometric iteration method can be found in [28-30]. They are simply referred to as the H-H-H method hereafter.

In order to improve efficiency, we propose the following geometric approximation by normal curvature. Vector $\boldsymbol{p} - \boldsymbol{p}_0$ can be expressed as a linear combination of the tangent vectors $\frac{\partial \boldsymbol{s}}{\partial u}$, $\frac{\partial \boldsymbol{s}}{\partial v}$ and the unit normal vector $\boldsymbol{n}$ at $\boldsymbol{p}_0$ which is actually the following formula (3),

$$\boldsymbol{p} - \boldsymbol{p}_0 = \alpha_1\frac{\partial \boldsymbol{s}}{\partial u} + \alpha_2\frac{\partial \boldsymbol{s}}{\partial v} + \alpha_3\boldsymbol{n}. \tag{3}$$

By simplifying, the unit normal vector can be specifically expressed as the following,

$$\boldsymbol{n} = \frac{\boldsymbol{A}}{\boldsymbol{B}} = \frac{(A_1,A_2,A_3)}{\sqrt{A_1^2+A_2^2+A_3^2}}, \tag{4}$$

where $A_1 = \frac{\partial f_2(u,v)}{\partial u}\frac{\partial f_3(u,v)}{\partial v} - \frac{\partial f_3(u,v)}{\partial u}\frac{\partial f_2(u,v)}{\partial v}$, $A_2 = \frac{\partial f_3(u,v)}{\partial u}\frac{\partial f_1(u,v)}{\partial v} - \frac{\partial f_1(u,v)}{\partial u}\frac{\partial f_3(u,v)}{\partial v}$ and $A_3 = \frac{\partial f_1(u,v)}{\partial u}\frac{\partial f_2(u,v)}{\partial v} - \frac{\partial f_2(u,v)}{\partial u}\frac{\partial f_1(u,v)}{\partial v}$. Multiplying both sides of (3) by $\frac{\partial \boldsymbol{s}}{\partial u}$, $\frac{\partial \boldsymbol{s}}{\partial v}$, $\boldsymbol{n}$, respectively, we obtain

$$\begin{cases} \alpha_1\langle\frac{\partial \boldsymbol{s}}{\partial u}, \frac{\partial \boldsymbol{s}}{\partial u}\rangle + \alpha_2\langle\frac{\partial \boldsymbol{s}}{\partial u}, \frac{\partial \boldsymbol{s}}{\partial v}\rangle + \alpha_3\langle\frac{\partial s}{\partial u}, \boldsymbol{n}\rangle \\ = \langle \boldsymbol{p}-\boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial u}\rangle, \\ \alpha_1\langle\frac{\partial \boldsymbol{s}}{\partial u}, \frac{\partial \boldsymbol{s}}{\partial v}\rangle + \alpha_2\langle\frac{\partial \boldsymbol{s}}{\partial v}, \frac{\partial \boldsymbol{s}}{\partial v}\rangle + \alpha_3\langle\frac{\partial s}{\partial v}, \boldsymbol{n}\rangle \\ = \langle \boldsymbol{p}-\boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial v}\rangle, \\ \alpha_1\langle\frac{\partial \boldsymbol{s}}{\partial u}, \boldsymbol{n}\rangle + \alpha_2\langle\frac{\partial \boldsymbol{s}}{\partial v}, \boldsymbol{n}\rangle + \alpha_3\langle \boldsymbol{n}, \boldsymbol{n}\rangle \\ = \langle \boldsymbol{p}-\boldsymbol{p}_0, \boldsymbol{n}\rangle. \end{cases} \tag{5}$$

The coefficients of the first fundamental form are given by $E = \langle\frac{\partial \boldsymbol{s}}{\partial u}, \frac{\partial \boldsymbol{s}}{\partial u}\rangle$, $F = \langle\frac{\partial \boldsymbol{s}}{\partial u}, \frac{\partial \boldsymbol{s}}{\partial v}\rangle$, $G = \langle\frac{\partial \boldsymbol{s}}{\partial v}, \frac{\partial \boldsymbol{s}}{\partial v}\rangle$. On the other hand, in order to conveniently express the derivation of the following formulas, the coefficients $\langle\frac{\partial \boldsymbol{s}}{\partial u}, \boldsymbol{n}\rangle$, $\langle\frac{\partial \boldsymbol{s}}{\partial v}, \boldsymbol{n}\rangle$, $\langle \boldsymbol{n}, \boldsymbol{n}\rangle$, $\langle \boldsymbol{p}-\boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial u}\rangle$, $\langle \boldsymbol{p}-\boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial v}\rangle$, $\langle \boldsymbol{p}-\boldsymbol{p}_0, \boldsymbol{n}\rangle$ can be simplified as $C_1 = \langle\frac{\partial \boldsymbol{s}}{\partial u}, \boldsymbol{n}\rangle$, $C_2 = \langle\frac{\partial \boldsymbol{s}}{\partial v}, \boldsymbol{n}\rangle$, $C_3 = \langle \boldsymbol{n}, \boldsymbol{n}\rangle = 1$, $S_1 = \langle \boldsymbol{p}-\boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial u}\rangle$, $S_2 = \langle \boldsymbol{p}-\boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial v}\rangle$, $S_3 = \langle \boldsymbol{p}-\boldsymbol{p}_0, \boldsymbol{n}\rangle$. By simplifying (5), it is easy to get

$$\begin{cases} \alpha_1E + \alpha_2F + \alpha_3C_1 = S_1, \\ \alpha_1F + \alpha_2G + \alpha_3C_2 = S_2, \\ \alpha_1C_1 + \alpha_2C_2 + \alpha_3C_3 = S_3. \end{cases} \tag{6}$$

By computing the solution of a regular system of linear equations in (6) about α_1, α_2 and α_3, we obtain the parameters α_1, α_2, α_3 as follows (notice $C_3 = 1$),

$$\begin{cases} \alpha_1 = -\frac{C_1C_2S_2 - C_1GS_3 - C_2^2S_1 + C_2FS_3 - FS_2 + GS_1}{C_1^2G - 2C_1C_2F + C_2^2E - EG + F^2}, \\ \alpha_2 = \frac{C_1^2S_2 - C_1C_2S_1 - C_1FS_3 + C_2ES_3 - ES_2 + FS_1}{C_1^2G - 2C_1C_2F + C_2^2E - EG + F^2}, \\ \alpha_3 = -\frac{C_1FS_2 - C_1GS_1 - C_2ES_2 + C_2FS_1 + EGS_3 - F^2S_3}{C_1^2G - 2C_1C_2F + C_2^2E - EG + F^2}. \end{cases} \tag{7}$$

According to the definition of normal curvature of differential geometry, if two curves on parametric surface $\boldsymbol{s}(u,v)$ at point $\boldsymbol{p}_0$ have the same unit tangent vector, then the two curves have the same normal curvature at point $\boldsymbol{p}_0$. In fact, it is not difficult to know that the tangent vector on surface $\boldsymbol{s}(u,v)$ at point $\boldsymbol{p}_0$ is $\boldsymbol{T} = \alpha_1\frac{\partial \boldsymbol{s}}{\partial u} + \alpha_2\frac{\partial \boldsymbol{s}}{\partial v}$, and then the corresponding unit tangent vector of $\boldsymbol{T}$ is $\frac{\boldsymbol{T}}{\|\boldsymbol{T}\|} = \frac{\alpha_1}{\|\boldsymbol{T}\|}\frac{\partial \boldsymbol{s}}{\partial u} + \frac{\alpha_2}{\|\boldsymbol{T}\|}\frac{\partial \boldsymbol{s}}{\partial v}$. Therefore the normal curvature of parametric surface $\boldsymbol{s}(u,v)$ along the unit tangent vector $\frac{\boldsymbol{T}}{\|\boldsymbol{T}\|}$ at point $\boldsymbol{p}_0$ can be defined as

$$k_n(\frac{\boldsymbol{T}}{\|\boldsymbol{T}\|}) = \frac{L\alpha_1^2 + 2M\alpha_1\alpha_2 + N\alpha_2^2}{\|\boldsymbol{T}\|^2}, \tag{8}$$

where L, M, N are the coefficients of the second fundamental form of parametric surface of differential geometry. More specifically, L, M, N can be defined as $L = \langle\frac{\partial^2 \boldsymbol{s}}{\partial u^2}, \boldsymbol{n}\rangle$, $M = \langle\frac{\partial^2 \boldsymbol{s}}{\partial u\partial v}, \boldsymbol{n}\rangle$, $N = \langle\frac{\partial^2 \boldsymbol{s}}{\partial v^2}, \boldsymbol{n}\rangle$, where

$$\begin{aligned} \frac{\partial^2 \boldsymbol{s}}{\partial u^2} &= (\frac{\partial^2 f_1(u,v)}{\partial u^2}, \frac{\partial^2 f_2(u,v)}{\partial u^2}, \frac{\partial^2 f_3(u,v)}{\partial u^2}), \\ \frac{\partial^2 \boldsymbol{s}}{\partial u\partial v} &= (\frac{\partial^2 f_1(u,v)}{\partial u\partial v}, \frac{\partial^2 f_2(u,v)}{\partial u\partial v}, \frac{\partial^2 f_3(u,v)}{\partial u\partial v}), \\ \frac{\partial^2 \boldsymbol{s}}{\partial v^2} &= (\frac{\partial^2 f_1(u,v)}{\partial v^2}, \frac{\partial^2 f_2(u,v)}{\partial v^2}, \frac{\partial^2 f_3(u,v)}{\partial v^2}). \end{aligned}$$

From the tangent vector $\boldsymbol{T}$, we get the following expression,

$$\begin{aligned} \|\boldsymbol{T}\|^2 &= \langle\alpha_1\frac{\partial \boldsymbol{s}}{\partial u} + \alpha_2\frac{\partial \boldsymbol{s}}{\partial v}, \alpha_1\frac{\partial \boldsymbol{s}}{\partial u} + \alpha_2\frac{\partial \boldsymbol{s}}{\partial v}\rangle \\ &= E\alpha_1^2 + 2F\alpha_1\alpha_2 + G\alpha_2^2, \end{aligned} \tag{9}$$

where the coefficients E, F, G are the first fundamental form of parametric surface of differential geometry. Based on (7)–(9), the normal curvature can be specifically expressed as

$$k_n(\frac{\boldsymbol{T}}{\|\boldsymbol{T}\|}) = \frac{II(\boldsymbol{T},\boldsymbol{T})}{I(\boldsymbol{T},\boldsymbol{T})} = \frac{L\alpha_1^2 + 2M\alpha_1\alpha_2 + N\alpha_2^2}{E\alpha_1^2 + 2F\alpha_1\alpha_2 + G\alpha_2^2}. \quad (10)$$

Here, let us make a few explanations about the uniqueness of the normal curvature of (10). According to the basic definition of normal curvature of differential geometry, countless values of the normal curvature can exist at a specified point $\boldsymbol{p}_0$ on a parametric surface $\boldsymbol{s}(u,v)$. But the normal curvature at this moment is unique along the tangential direction of the current normal transversal curve at the initial iterative point $\boldsymbol{p}_0$. And the current normal transversal curve is created when the current normal section plane Π determined by the unit normal vector $\boldsymbol{n}$ at the initial iterative point $\boldsymbol{p}_0$ and the vector $\overrightarrow{\boldsymbol{p}-\boldsymbol{p}_0}$ intersects the parametric surface $\boldsymbol{s}(u,v)$ (see Fig.2(b)).

In the following, we present the iterative formula for computing parameter increment Δu, Δv determined by normal curvature $k_n(\frac{\boldsymbol{T}}{|\boldsymbol{T}|})$ or k_n. The radius R and center $\boldsymbol{m}$ of normal curvature sphere of parametric surface $\boldsymbol{s}(u,v)$ at point $\boldsymbol{p}_0$ can be represented by the following formulas respectively.

$$R = \left|\frac{1}{k_n}\right|. \quad (11)$$

$$\boldsymbol{p}_0 + \frac{\boldsymbol{n}}{k_n} = \boldsymbol{m}. \quad (12)$$

From (11) and (12), we can obtain the equation of normal curvature sphere:

$$\|\boldsymbol{x} - \boldsymbol{m}\| = R, \quad (13)$$

where $\boldsymbol{x} = (x,y,z)$. On the other hand, the parametric equation of the line segment connecting the test point $\boldsymbol{p}$ and the center $\boldsymbol{m}$ of normal curvature sphere (13) can be expressed as,

$$\boldsymbol{x} = \boldsymbol{p} + (\boldsymbol{m} - \boldsymbol{p})w, \quad (14)$$

where w $(0 < w < 1)$ will be an undetermined parameter. Since the footpoint $\boldsymbol{q} = (q_1, q_2, q_3)$ is the intersection of the line segment (14) and the normal curvature sphere (13), substituting (14) into (13), the undetermined parameter w could be specifically expressed as

$$w = 1 \pm \frac{R}{\|\boldsymbol{m} - \boldsymbol{p}\|}. \quad (15)$$

Since the footpoint $\boldsymbol{q}$ lies between the line segment $\overline{\boldsymbol{mp}}$ determined by the center $\boldsymbol{m}$ of the normal curvature sphere and the test point $\boldsymbol{p}$, the parameter w is in $(0,1)$. By (15), the parameter w should be

$$w = 1 - \frac{R}{\|\boldsymbol{m} - \boldsymbol{p}\|}. \quad (16)$$

This time, the footpoint $\boldsymbol{q}$ can be expressed as

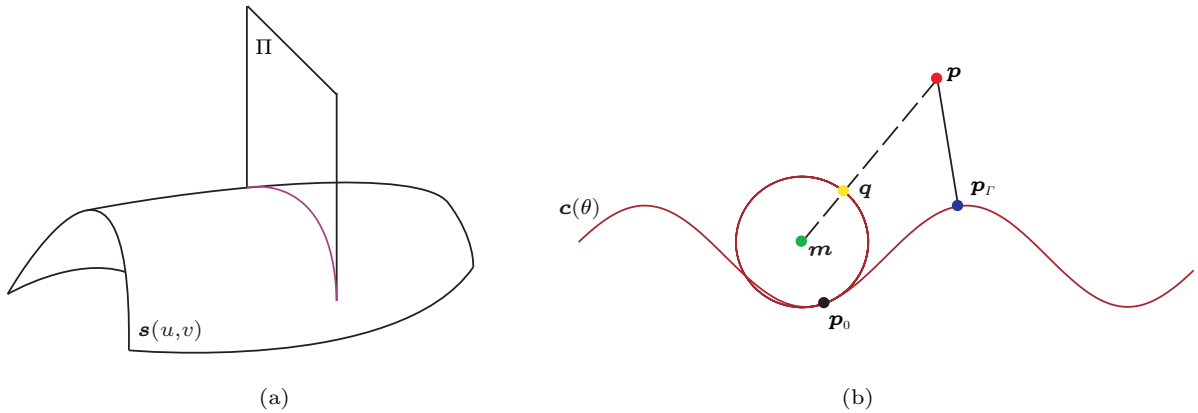
$$\boldsymbol{q} = \boldsymbol{p} + (\boldsymbol{m} - \boldsymbol{p})w. \quad (17)$$

Analogous to the first-order geometric iteration method[28–30], we can get the most core iterative formula associated with the GSA,

$$\boldsymbol{q} - \boldsymbol{p}_0 \approx \frac{\partial \boldsymbol{s}}{\partial u}\Delta u + \frac{\partial \boldsymbol{s}}{\partial v}\Delta v. \quad (18)$$

Through multiplying both sides of (18) by $\frac{\partial \boldsymbol{s}}{\partial u}$, $\frac{\partial \boldsymbol{s}}{\partial v}$, respectively, we have the iterative form:

$$\begin{cases} E\Delta u + F\Delta v = C_4, \\ F\Delta u + G\Delta v = C_5, \end{cases} \quad (19)$$



(a) (b)

Fig.2. Entire graphic demonstration of convergence analysis for GSA. (a) The normal section plane Π intersects the surface $\boldsymbol{s}(u,v)$ of a normal transversal curve. (b) The normal section plane Π traverses the current normal curvature sphere to form a curvature circle.

where $C_4 = \langle \boldsymbol{q} - \boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial u} \rangle$, $C_5 = \langle \boldsymbol{q} - \boldsymbol{p}_0, \frac{\partial \boldsymbol{s}}{\partial v} \rangle$. Solving the system of linear equations in (19), we can get the iterative increment formula associated with the GSA:

$$\begin{cases} \Delta u = \dfrac{C_4 G - C_5 F}{EG - F^2}, \\ \Delta v = -\dfrac{C_4 F - C_5 E}{EG - F^2}. \end{cases} \tag{20}$$

We now update u_0 and v_0 according to $u_0 + \Delta u \to u_0$, $v_0 + \Delta v \to v_0$, and the procedure (20) is repeated again, with $\boldsymbol{s}(u_0, v_0)$ as a new initial point, until u_0 and v_0 make the desired accuracy criteria being met. Namely, we increase u_0, v_0 by Δu, Δv respectively and repeat the above procedure (20) until Δu and Δv are less than a given tolerance ($|\Delta u| < \varepsilon$ and $|\Delta v| < \varepsilon$ or they satisfy the inequality $|\Delta u|^2 + |\Delta v|^2 < \varepsilon$). In this way, we can compute the orthogonal projection point $\boldsymbol{p}_\Gamma$ of test point $\boldsymbol{p}$ onto the surface (see Fig.1). The algorithm with the geometric strategy can be realized as Algorithm 1.

Algorithm 1. Geometric Strategy Algorithm

Input: the initial parametric value $\boldsymbol{t}_0 = (u_0, v_0)^{\mathrm{T}}$, the parametric surface $\boldsymbol{s}(u, v)$ and test point $\boldsymbol{p}$

Output: the final iterative parametric value $\boldsymbol{t}_n = (u_n, v_n)^{\mathrm{T}}$ or the orthogonal projection point $\boldsymbol{p}_\Gamma$

Step 1. Input the initial iterative parametric value $\boldsymbol{t}_0$.

Step 2. Use (20), compute the parametric incremental value $\Delta \boldsymbol{t} = (\Delta u, \Delta v)^{\mathrm{T}}$, and update $\boldsymbol{t}_0 + \Delta \boldsymbol{t}$ to $\boldsymbol{t}_0$, namely, $\boldsymbol{t}_0 = \boldsymbol{t}_0 + \Delta \boldsymbol{t}$.

Step 3. Judge whether the norm of difference between the former $\boldsymbol{t}_0$ and the latter $\boldsymbol{t}_0$ is near 0 ($\|\Delta \boldsymbol{t}\| < \varepsilon$). If so, end this algorithm; if not, go to step 2.

Remark 1. We give a geometric interpretation of GSA in Fig.1, where $\boldsymbol{s}(u, v)$ is a parameter surface, $\boldsymbol{p}_0 = \boldsymbol{s}(u_0, v_0)$ with the initial iterative parameter (u_0, v_0) on the parametric surface $\boldsymbol{s}(u, v)$ is the initial iterative point. According to the basic characteristic of elementary differential geometry, the initial iterative point $\boldsymbol{p}_0 = \boldsymbol{s}(u_0, v_0)$ and the vector $\overrightarrow{\boldsymbol{p} - \boldsymbol{p}_0}$ determine a unique normal curvature (defined by (10)). Consequently, the unique normal curvature sphere with the radius R and the center $\boldsymbol{m}$ is determined by (11)–(13). The point $\boldsymbol{q}$ is the intersection of the line segment $\overline{\boldsymbol{pm}}$ and the normal curvature sphere. $\boldsymbol{p}_\Gamma$ is the corresponding orthogonal projection point of the test point $\boldsymbol{p}$ onto the parametric surface $\boldsymbol{s}(u, v)$. u_0 and v_0 are updated according to $u_0 + \Delta u \to u_0$, and $v_0 + \Delta v \to v_0$ respectively. The procedure (20) is repeated with $\boldsymbol{s}(u_0, v_0)$ as a new initial point until u_0 and v_0 make the desired accuracy criteria being met.

2.2 Processing Degenerate Status with $k_n = 0$

It is well known that the most important iterative increment formula associated with GSA is the iterative formula (20). If the normal curvature of the iterative formula (10) is zero, the whole iteration process will degenerate. In order to solve this special degeneration, we adopt a small perturbation of normal curvature of (10) in programming implementation of GSA. Namely, the normal curvature of (10), k_n, could be incremented by a small positive constant ε, i.e., $k_n = k_n + \varepsilon$, and the iteration in (20) continues to compute the parametric incremental value $\Delta \boldsymbol{t} = (\Delta u, \Delta v)^{\mathrm{T}}$, and update $\boldsymbol{t}_0 + \Delta \boldsymbol{t}$ to $\boldsymbol{t}_0$. Of course, this special degeneration status seldom appears in the actual programming implementation of GSA. If this kind of special degeneration occurs, we will try to find all the plane patches or line segments of the parametric surface $\boldsymbol{s}(u, v)$. We then directly seek the point (written as $\boldsymbol{p}_{0\_\Gamma}$) with the minimum distance between the test point $\boldsymbol{p}$ and all the plane patches or the line segments of the parametric surface $\boldsymbol{s}(u, v)$. After that, we orthogonally project the test point $\boldsymbol{p}$ onto the parametric surface with the elimination of all patches or line segments by using GSA, where the corresponding orthogonal projection point is written as $\boldsymbol{p}_\Gamma$. Then, from two points $\boldsymbol{p}_{0\_\Gamma}$ and $\boldsymbol{p}_\Gamma$, a corresponding point is selected such that the distance between the corresponding point and the test point $\boldsymbol{p}$ is the minimum one. If the entire program terminates, the minimum distance and its corresponding parameter value are acquired. Therefore this avoids repeated small perturbation by using (10) directly.

3 Convergence Analysis

In this section, we consider the convergence analysis of iterative technique given in (20). We try to prove that GSA is independent of the initial iterative value.

Theorem 1. *GSA is independent of the initial iterative value.*

Proof. The proof is divided into two parts, the first part is the analysis and proof of the case with the corresponding unique orthogonal projection point for the test point $\boldsymbol{p}$ and the second part is the analysis and proof of the case with the corresponding multiple orthogonal projection points for the test point $\boldsymbol{p}$.

Part 1. Firstly, we give a geometric interpretation of GSA in Fig.1 and Fig.2. From Fig.1, the line segment $\overline{\boldsymbol{pp}_\Gamma}$ and the line segment $\overline{\boldsymbol{pm}}$ span into a normal section plane Π (see Fig.2(a)). It is not hard to find that the initial iterative point $\boldsymbol{p}_0$ is also on the

normal section plane Π. Based on the basic characteristic of differential geometry, the normal section plane Π intersects the surface $\boldsymbol{s}(u,v)$ of a normal transversal curve (the brown curve of Fig.2(a) which is the normal transversal curve is exactly the same as the brown curve of Fig.2(b)). And the normal section plane Π traverses the current normal curvature sphere to form a curvature circle in Fig.2(b) which has the same center $\boldsymbol{m}$ and radius with the normal curvature sphere. Red, black, green, yellow and blue points represent the test point $\boldsymbol{p}$, the initial iterative point $\boldsymbol{p}_0$, the center $\boldsymbol{m}$ of curvature circle, the intersection $\boldsymbol{q}$ of the line segment $\overline{\boldsymbol{pm}}$ and the curvature circle, and the orthogonal projecting point $\boldsymbol{p}_\Gamma$, respectively (see Fig.2(b)).

Secondly, we interpret the geometric meaning of the algorithm described in Fig.2(b). There is a parametric curve $\boldsymbol{c}(\theta)$ defined by the normal transversal curve which is the intersection determined by the normal section plane Π, the surface $\boldsymbol{s}(u,v)$ and a test point $\boldsymbol{p}$, where the parameter θ is associated with two parameters u and v of the parametric surface $\boldsymbol{s}(u,v)$. We parameterize the curvature circle $\overline{\boldsymbol{c}}$ (see Fig.2(b)) such that it shares the same Taylor's polynomial with the parametric curve $\boldsymbol{c}(\theta)$. Then this equality holds:

$$\begin{aligned}\boldsymbol{q} &= \overline{\boldsymbol{c}}(\theta_0+\Delta\theta)\\ &= \boldsymbol{c}(\theta_0+\Delta\theta)+o(\Delta\theta^2)\\ &= \boldsymbol{c}(\theta_0)+\Delta\theta\boldsymbol{c}'(\theta_0)+\frac{\Delta\theta^2}{2}\boldsymbol{c}''(\theta_0)+o(\Delta\theta^2). \quad (21)\end{aligned}$$

In $\mathbb{R}^2$, we deal with (21) using the method in [18, 33] and get $\det(\boldsymbol{q}-\boldsymbol{c}(\theta_0),\boldsymbol{c}''(\theta_0))=\Delta\theta\det(\boldsymbol{c}'(\theta_0),\boldsymbol{c}''(\theta_0))+o(\Delta\theta^2)$, which yields the iterative formula

$$\begin{aligned}\Delta\theta &= \frac{\det(\boldsymbol{q}-\boldsymbol{c}(\theta_0),\boldsymbol{c}''(\theta_0))}{\det(\boldsymbol{c}'(\theta_0),\boldsymbol{c}''(\theta_0))}\\ &= \frac{1}{k\left\|\boldsymbol{c}'\right\|^3}\det(\boldsymbol{q}-\boldsymbol{c}(\theta_0),\boldsymbol{c}''(\theta_0)), \quad (22)\end{aligned}$$

where k is the curvature of curvature circle at $\theta=\theta_0$. θ is updated by $\theta_0+\Delta\theta\to\theta_0$ and the procedure (22) is repeated with $\boldsymbol{c}(\theta)$ as a new initial point until θ_0 meets the desired accuracy criteria. This is the classic curvature circle method which orthogonally projects the test point onto the planar parametric curve[25].

From the above description of the transformation process, GSA in essence is equivalent to the classic curvature circle method for the point projection problem. Since Theorem 3 in [27] has proved that the classic curvature circle method[25] is independent of the initial iteration value, GSA is also independent of the initial iterative value.

Part 2. For the case with corresponding multiple orthogonal projection points of the test point $\boldsymbol{p}$, let us assume that there are n corresponding orthogonal projection points $\boldsymbol{p}_{1\_\Gamma}$, $\boldsymbol{p}_{2\_\Gamma}$, ..., $\boldsymbol{p}_{n\_\Gamma}$. According to the most essential geometric characteristics of GSA, once the initial iteration point $\boldsymbol{p}_0=\boldsymbol{s}(u_0,v_0)$ is determined, the corresponding unique orthogonal projection point must be satisfied with the proximity principle such that the two points distance function $\|\boldsymbol{p}_0-\boldsymbol{p}_{i\_\Gamma}\|\,(i=1,2,...,n)$ must be minimal. Then corresponding unique orthogonal projection point is also be determined by the initial iteration point $\boldsymbol{p}_0$. Namely, once the initial iterative point $\boldsymbol{p}_0$ is determined by two parameters u_0 and v_0, the initial iterative point $\boldsymbol{p}_0$ can always find a unique orthogonal projection point in all n corresponding orthogonal projection points $\boldsymbol{p}_{1\_\Gamma}$, $\boldsymbol{p}_{2\_\Gamma}$, ..., $\boldsymbol{p}_{n\_\Gamma}$ such that the distance between the current initial iterative point $\boldsymbol{p}_0$ and the current orthogonal projection point must be minimal. This conclusion indicates that GSA is also independent of the initial iterative value for the corresponding multiple orthogonal projection points of the test point $\boldsymbol{p}$. □

Remark 2. In Theorem 1, we prove that GSA is independent of the initial value. In addition, if the test point $\boldsymbol{p}$ is not on the parametric surface $\boldsymbol{s}(u,v)$, it is not difficult to find that the convergence order of GSA is 1. On the contrary, if the test point $\boldsymbol{p}$ is on the parametric surface $\boldsymbol{s}(u,v)$, it is not hard to demonstrate that the convergence order of GSA is 2. The proof method is analogous to that of Theorem 1 in [34].

4 Experimental Results

In order to explain the advantage of GSA to other algorithms (the H-H-H method, the Newton's method, etc.), we present two numerical examples to verify its robustness and high efficiency. From Tables 1–10, we can find that the iterative termination criterion is satisfied with two conditions $(u_{n+1}-\alpha)^2+(v_{n+1}-\beta)^2<1\mathrm{E}-14$ and $(u_{n+1}-u_n)^2+(v_{n+1}-v_n)^2<1\mathrm{E}-14$. All numerical results were computed through g++ in a Fedora Linux 8 environment. The approximate zero is reached up to the 14th decimal place. These results of our two examples are obtained from the computer hardware configuration with T2080 1.73 GHz CPU and 2.5 GB memory. In the following 10 tables of the two examples, the test time is measured in nanosecond.

Example 1. We consider the surface $\boldsymbol{s}(u,v)=(u+v,\sin(u)+2\cos(v),\sin(u+v))$ with a test point $\boldsymbol{p}=(p_1,p_2,p_3)=(1.0,2.0,2.0)$. Using GSA, the corresponding orthogonal projection parametric point

is (1.289 024 00, 2.000 000 00, 0.960 564 136), and the initial iterative values (u_0, v_0) are $(23, 23)$, $(23, -23)$, $(-23, 22)$, $(-23, -23)$, $(15, 15)$, $(15, -15)$, $(-15, 14)$, $(-13, -15)$, respectively. Each initial iterative value repeatedly iterates 10 times, respectively, and yields 10 different iteration time in nanosecond. In Table 1, the mean running time of GSA is 69 077, 11 113, 45 886, 67 531, 67 453, 11 632, 43 421, 63 451 nanoseconds for eight different initial iterative values, respectively. In the end, the totally average running time in Table 1 is 47 445 nanoseconds (= 0.047 445 ms). Using the H-H-H method, the corresponding orthogonal projection parametric point and the initial iterative values (u_0, v_0) remain the same. Using the same iteration frequency, we also get 10 different iteration time in nanosecond. In Table 2, the mean running time of the H-H-H method is 214 328, 195 087, 209 855, 178 796, 172 946, 181 253, 172 951, 172 811 nanoseconds for eight different initial iterative values, respectively. In the end, the average running time in Table 2 is 187 253 nanoseconds (= 0.187 253 ms). We then change the corresponding orthogonal projection parametric points with value of (1.289 024 00, 2.000 000 00, 0.960 564 136), and the initial iterative values (u_0, v_0) are set as $(5, 5)$, $(5, -5)$, $(-4, 5)$, $(-5, -5)$, $(2, 2)$, $(2, -2)$, $(-2, 3)$, $(-2, -2)$, respectively, while the statistical method to generate the iteration time is exactly the same as that in Table 1 and Table 2. The new results of GSA and the H-H-H method are reported. The mean running time of GSA is 50 188, 10 996, 42 344, 56 155, 48 388, 11 189, 48 733, 45 027 nanoseconds for eight different initial iterative values, respectively. As a result, the average running time of GSA is 39 128 nanoseconds (= 0.039 128 ms). Using the H-H-H method, the corresponding orthogonal projection parametric point is also (1.289 024 00, 2.000 000 00, 0.960 564 136) and the initial iterative values (u_0, v_0) are $(5, 5)$, $(5, -5)$, $(-4, 5)$, $(-5, -5)$, $(2, 2)$, $(2, -2)$, $(-2, 3)$, $(-2, -2)$, respectively. Each initial iterative value repeatedly iterates 10 times, respectively, and yields 10 different iteration time with the time unit of nanosecond. The mean running time of the H-H-H method is 176 507, 174 899, 177 187, 179 365, 170 479, 208 636, 205 302, 179 656 nanoseconds for eight different initial iterative values, respectively. As a result, the overall average running time of the H-H-H method is 184 004 nanoseconds (= 0.184 004 ms). Table 3 shows the results of different methods with different initial iterative values (u_0, v_0). In this table, NC means that it cannot converge to the needed root. From Table 3, it can be found that the H-H-H method and GSA have better convergence and robustness, while Newton's second-order method is dependent on the initial iterative value and unstable. Furthermore, GSA converges faster than the H-H-H method. Table 4 and Table 5 adopt parametric incremental iterative procedure with increment of $(\Delta u, \Delta v)$ for the H-H-H method and GSA where the test point is $\boldsymbol{p} = (1, 2, 2)$ and the initial parametric value is $(1, 2)$. To sum up, although the

Table 1. Running Time (ns) for Different Initial Iterative Values by GSA

(u_0, v_0)	1	2	3	4	5	6	7	8	9	10	Average	Total Average
(23, 23)	67 896	68 157	67 948	68 587	68 359	69 655	69 883	70 038	69 732	70 515	69 077	47 445
(23, –23)	11 396	11 169	10 938	11 208	11 149	11 076	11 151	11 245	11 310	10 483	11 113	
(–23, 22)	46 384	46 644	45 677	45 679	45 975	46 364	44 970	45 722	46 097	45 345	45 886	
(–23, –23)	67 080	67 328	67 520	67 443	67 443	67 711	67 476	67 683	66 731	68 893	67 531	
(15, 15)	68 042	67 780	67 126	67 998	68 144	67 674	67 259	67 257	65 363	67 891	67 453	
(15, –15)	11 603	11 624	11 682	11 686	11 684	11 493	11 607	11 720	11 672	11 551	11 632	
(–15, 14)	44 025	41 123	42 481	42 252	42 472	43 481	44 086	43 824	45 453	45 017	43 421	
(–13, –15)	62 899	63 333	63 598	62 946	63 172	63 440	64 062	63 648	63 190	64 217	63 451	

Table 2. Running Time (ns) for Different Initial Iterative Values by the H-H-H Method

(u_0, v_0)	1	2	3	4	5	6	7	8	9	10	Average	Total Average
(23, 23)	237 547	221 936	224 999	223 054	222 835	178 692	220 057	179 441	202 800	231 916	214 328	187 253
(23, –23)	184 946	185 304	185 252	235 680	184 976	184 932	184 923	184 783	184 805	235 270	195 087	
(–23, 22)	246 152	235 524	208 764	187 238	237 375	230 656	187 015	191 404	187 265	187 161	209 855	
(–23, –23)	171 961	185 132	186 333	185 534	185 930	186 408	172 052	171 674	171 325	171 615	178 796	
(15, 15)	172 908	173 107	172 917	172 947	172 914	172 935	172 940	172 941	172 924	172 929	172 946	
(15, –15)	171 122	171 064	171 066	212 833	171 165	171 071	230 814	171 196	171 067	171 129	181 253	
(–15, 14)	172 963	172 977	173 059	172 901	172 909	172 875	172 881	172 975	173 001	172 964	172 951	
(–13, –15)	170 647	177 460	170 546	170 727	170 557	170 538	170 551	170 545	170 571	185 964	172 811	

H-H-H method and GSA are both first-order convergence, the number of iterations for GSA is significantly less than that for the H-H-H method, e.g., GSA performs far superior (see Fig.3).

Table 3. Comparison of Number of Iteration for Three Iterative Methods with Different Initial Parametric Values

(u_0, v_0)	H-H-H Method	GSA	Newton's Method
(1.0, 2.0)	282	29	NC
(2.0, 0.5)	284	33	NC
(–1.0, –2.5)	288	32	NC
(1.0, –0.5)	284	31	NC
(–0.3, –0.5)	288	26	NC
(–1.0, 1.1)	283	30	NC
(–0.7, 0.4)	284	30	NC

Table 4. Parametric Incremental Iterative Procedure According to $(\Delta u, \Delta v)$ of the H-H-H Method with the Test Point $\boldsymbol{p} = (1, 2, 2)$ and the Initial Parametric Value $(1, 2)$ in Example 1

Step	Δu	Δv
1	-6.51×10^{-1}	-1.29×10^{0}
10	1.02×10^{-1}	6.40×10^{-2}
20	4.48×10^{-2}	2.83×10^{-2}
30	2.07×10^{-2}	1.31×10^{-2}
40	9.64×10^{-3}	6.09×10^{-3}
50	4.50×10^{-3}	2.85×10^{-3}
60	2.10×10^{-3}	1.33×10^{-3}
70	9.84×10^{-4}	6.22×10^{-4}
80	4.60×10^{-4}	2.91×10^{-4}
90	2.15×10^{-4}	1.36×10^{-4}
100	1.01×10^{-4}	6.36×10^{-5}
110	4.70×10^{-5}	2.97×10^{-5}
120	2.20×10^{-5}	1.39×10^{-5}
130	1.03×10^{-5}	6.50×10^{-6}
140	4.80×10^{-6}	3.04×10^{-6}
150	2.25×10^{-6}	1.42×10^{-6}
160	1.05×10^{-6}	6.64×10^{-7}
170	4.91×10^{-7}	3.11×10^{-7}
180	2.30×10^{-7}	1.45×10^{-7}
190	1.07×10^{-7}	6.79×10^{-8}
200	5.02×10^{-8}	3.17×10^{-8}
210	2.35×10^{-8}	1.48×10^{-8}
220	1.10×10^{-8}	6.94×10^{-9}
230	5.13×10^{-9}	3.24×10^{-9}
240	2.40×10^{-9}	1.52×10^{-9}
250	1.12×10^{-9}	7.09×10^{-10}
260	5.24×10^{-10}	3.32×10^{-10}
270	2.45×10^{-10}	1.55×10^{-10}
275	-1.67×10^{-10}	-1.06×10^{-10}
280	1.15×10^{-10}	7.25×10^{-11}
281	-1.06×10^{-10}	-6.72×10^{-11}
282	9.84×10^{-11}	6.22×10^{-11}

Table 5. Parametric Incremental Iterative Procedure According to $(\Delta u, \Delta v)$ of the GSA with the Test Point $\boldsymbol{p} = (1, 2, 2)$ and the Initial Parametric Value $(1, 2)$ in Example 1

Step	Δu	Δv
1	-6.50×10^{-1}	-1.29×10^{0}
2	1.40×10^{-1}	4.69×10^{-2}
3	2.89×10^{-2}	-2.89×10^{-3}
4	1.48×10^{-2}	-5.75×10^{-3}
5	8.15×10^{-3}	-5.62×10^{-3}
6	4.13×10^{-3}	-3.84×10^{-3}
7	1.96×10^{-3}	-1.96×10^{-3}
8	9.19×10^{-4}	-9.19×10^{-4}
9	4.30×10^{-4}	-4.30×10^{-4}
10	2.01×10^{-4}	2.01×10^{-4}
11	9.39×10^{-5}	-9.39×10^{-5}
12	4.39×10^{-5}	-4.39×10^{-5}
13	2.05×10^{-5}	-2.05×10^{-5}
14	9.57×10^{-6}	-9.57×10^{-6}
15	4.47×10^{-6}	-4.47×10^{-6}
16	2.09×10^{-6}	-2.09×10^{-6}
17	9.76×10^{-7}	-9.76×10^{-7}
18	4.56×10^{-7}	4.56×10^{-7}
19	2.13×10^{-7}	-2.13×10^{-7}
20	9.96×10^{-8}	-9.96×10^{-8}
21	4.65×10^{-8}	-4.65×10^{-8}
22	2.17×10^{-8}	-2.17×10^{-8}
23	1.02×10^{-8}	-1.02×10^{-8}
24	4.75×10^{-9}	-4.75×10^{-9}
25	2.22×10^{-9}	-2.22×10^{-9}
26	1.04×10^{-9}	1.04×10^{-9}
27	4.84×10^{-10}	-4.84×10^{-10}
28	2.26×10^{-10}	-2.26×10^{-10}
29	1.06×10^{-10}	-1.06×10^{-10}

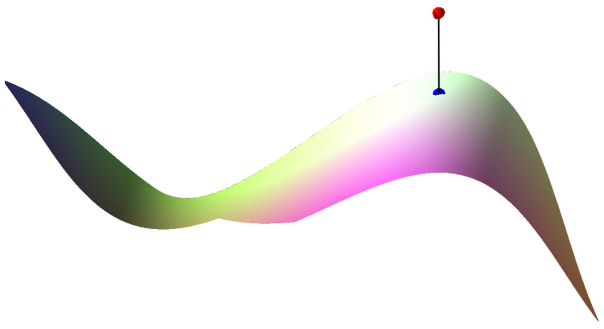


Fig.3. Graphic demonstration for example 1.

Remark 3. From the result of example 1, the overall average running time of GSA is 43.29 μs. The overall average running time of the H-H-H method[28–30] is

185.63 μs. From the results of example 1 and example 2 in [24], the overall average running time of the algorithm of [24] is 336.22 μs. From the results in [25], the overall average running time of the algorithm of [25] is 379.36 μs. From the results of three examples in [22], the overall average running time of the algorithm of [22] is 293.84 μs. From the results in [1], the overall average running time of the algorithm of [1] is 418.57 μs. From the results in [13], the overall average running time of the algorithm of [13] is 514.23 μs. From the results of the third line of Table 2 in [35], the overall average running time of the algorithm of [35] is 425.34 μs. Table 6 shows the time comparison for these algorithms. In short, the robustness and the efficiency of GSA are superior to those of the existing algorithms[1,13,22,24,25,28−30,35].

Table 6. Time Comparison of Various Algorithms with Example 1

Algorithm	Time (μs)
GSA	43.29
H-H-H method[28−30]	185.63
Algorithm in [24]	336.22
Algorithm in [25]	379.36
Algorithm in [22]	293.84
Algorithm in [1]	418.57
Algorithm in [13]	514.23
Algorithm in [35]	425.34

Example 2. We consider the surface $\boldsymbol{s}(u,v) = (u^4v^3 + uv + u, u^3 + v^3, u + v)$ with a test point $\boldsymbol{p} = (p_1, p_2, p_3) = (3.0, 4.0, 5.0)$. If the initial iterative values (u_0, v_0) take one of $(5,5)$, $(5,-5)$, $(-4,5)$, $(-5,-5)$, $(2,2)$, $(2,-2)$, $(-2,3)$ and $(-2,-2)$, the corresponding orthogonal projection parametric value and the corresponding orthogonal projection point will be (0.761 843 756 757 090, 1.562 910 200 608 70) and (3.238 610 455 103 38, 4.259 881 072 120 84, 2.324 753 957 365 79), respectively. The statistical method to generate iteration time is exactly the same as that in Table 1 and Table 2. For each initial iterative value, the procedure repeats 10 times, yielding 10 iteration time. The average running time of GSA, the H-H-H method and the Newton's method are 20 341, 27 769 and 74 911 nanoseconds, respectively. Table 7 compares the running time for various algorithms in Table 6, including the GSA, the H-H-H method and the Newton's method. GSA is faster than the H-H-H method and the Newton's method. Both the classic Newton's method and GSA can solve the system of nonlinear equations and need to calculate the second-order derivatives. The advantage of GSA is its independence of the initial iterative value, while the Newton's method is dependent on the initial iterative value and is locally convergent. Therefore, in the case of point orthogonal projection onto parametric surface, GSA is superior to the Newton's method in terms of the robustness and the efficiency. In the same way, Table 7 verifies once again that the convergence rate of GSA method is faster than those of the existing methods[1,13,22,24,25,28−30,32,35], including the classic Newton's method. Therefore, the robustness and the efficiency of GSA are better than those of existential methods including the classic Newton's method and the H-H-H method. Table 8, Table 9 and Table 10 adopt parametric incremental iterative procedure with the increment of $(\Delta u, \Delta v)$ for the H-H-H method, the Newton's method and GSA where the test point is $\boldsymbol{p} = (3,4,5)$ and the initial value is $(2,-2)$. The H-H-H method needs 25 iterations and the Newton's method needs 35 iterations, while GSA only needs 20 iterations. Although the H-H-H method and GSA are all first-order convergent, GSA saves five iterations less than the H-H-H method. Moreover, the Newton's method is second-order convergent, with the largest number of iterations among the three methods. Once more it is shown that GSA is superior to the H-H-H method and the Newton's method (see Fig.4).

Remark 4. Thanks to the reviewers' insightful comments, this remark is added to make a clear comparison between GSA and the method in [25]. The two methods share a great similarity, but they are very different in the parametric incremental iterative formula. For this reason, we give a detailed explanation.

Table 7. Time Comparison of Various Algorithms with Example 2

Algorithm	Time (μs)
GSA	20.34
H-H-H method[28−30]	27.77
Algorithm in [24]	336.22
Algorithm in [25]	379.36
Newton's method	74.91
Algorithm in [22]	293.84
Algorithm in [1]	418.57
Algorithm in [13]	514.23
Algorithm in [35]	425.34
Algorithm in [32]	237.90

Table 8. Parametric Incremental Iterative Procedure According to $(\Delta u, \Delta v)$ of the H-H-H Method with the Test Point $\boldsymbol{p} = (3, 4, 5)$ and the Initial Parametric Value $(2, -2)$ in Example 2

Step	Δu	Δv
1	-1.38×10^{-1}	5.03×10^{-1}
2	-1.25×10^{-1}	4.03×10^{-1}
3	-7.54×10^{-2}	3.79×10^{-1}
4	-2.79×10^{-2}	4.00×10^{-1}
5	-2.52×10^{-2}	7.64×10^{-1}
6	-1.71×10^{-2}	1.04×10^{-1}
7	-1.77×10^{-4}	-2.69×10^{-2}
8	-8.13×10^{-5}	3.36×10^{-3}
9	1.14×10^{-5}	-5.81×10^{-4}
10	-1.96×10^{-6}	9.73×10^{-5}
11	3.29×10^{-7}	-1.64×10^{-5}
12	-5.54×10^{-8}	2.75×10^{-6}
13	9.31×10^{-9}	-4.63×10^{-7}
14	-1.57×10^{-9}	7.79×10^{-8}
15	2.63×10^{-10}	-1.31×10^{-8}
16	-4.43×10^{-11}	2.20×10^{-9}
17	7.45×10^{-12}	-3.71×10^{-10}
18	-1.25×10^{-12}	6.23×10^{-11}
19	2.11×10^{-13}	-1.05×10^{-11}
20	-3.53×10^{-14}	1.76×10^{-12}
21	5.90×10^{-15}	-2.96×10^{-13}
22	-1.20×10^{-15}	4.98×10^{-14}
23	1.11×10^{-16}	-8.38×10^{-15}
24	8.90×10^{-17}	1.36×10^{-15}
25	1.48×10^{-16}	-2.71×10^{-16}
26	0	0

The corresponding formulas of (6) and (7) in [25] are the following two formulas,

$$area(\boldsymbol{c}', \boldsymbol{q} - \boldsymbol{c}(t_0)) = \frac{\Delta t^2}{2} area(\boldsymbol{c}', \boldsymbol{c}'') + o(\Delta t^2), \quad (23)$$

and

$$\begin{aligned} \Delta t^2 &\approx 2\frac{area(\boldsymbol{c}', \boldsymbol{q} - \boldsymbol{c}(t_0))}{area(\boldsymbol{c}', \boldsymbol{c}'')} \\ &= 2\frac{area(\boldsymbol{c}', \boldsymbol{q} - \boldsymbol{c}(t_0))}{k \left\| \boldsymbol{c}' \right\|^3}, \end{aligned} \quad (24)$$

where $area(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} \times \boldsymbol{y}\|$ and $area(\boldsymbol{x}, \boldsymbol{y})^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle \langle \boldsymbol{y}, \boldsymbol{y} \rangle - \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2$. From (24), we can get

$$\begin{aligned} &\Delta t^2 \\ &= 2\frac{area(\boldsymbol{c}', \boldsymbol{q} - \boldsymbol{c}(t_0))}{k \left\| \boldsymbol{c}' \right\|^3} \\ &= 2\frac{\langle \boldsymbol{c}', \boldsymbol{c}' \rangle \langle \boldsymbol{q} - \boldsymbol{c}(t_0), \boldsymbol{q} - \boldsymbol{c}(t_0) \rangle - \langle \boldsymbol{c}', \boldsymbol{q} - \boldsymbol{c}(t_0) \rangle^2}{k \left\| \boldsymbol{c}' \right\|^3}. \end{aligned} \quad (25)$$

Table 9. Parametric Incremental Iterative Procedure According to $(\Delta u, \Delta v)$ of the Newton's Method with the Test Point $\boldsymbol{p} = (3, 4, 5)$ and the Initial Parametric Value $(2, -2)$ in Example 2

Step	Δu	Δv
1	1.62×10^{-1}	-1.50×10^{-1}
2	1.56×10^{-1}	-1.33×10^{-1}
3	1.57×10^{-1}	-1.11×10^{-1}
4	1.75×10^{-1}	-6.95×10^{-2}
5	2.51×10^{-1}	5.23×10^{-2}
6	-5.59×10^{0}	-9.17×10^{0}
7	5.14×10^{-1}	5.84×10^{-1}
8	4.75×10^{-1}	5.39×10^{-1}
9	4.38×10^{-1}	4.97×10^{-1}
10	4.05×10^{-1}	4.59×10^{-1}
11	3.74×10^{-1}	4.24×10^{-1}
12	3.45×10^{-1}	3.91×10^{-1}
13	3.18×10^{-1}	3.61×10^{-1}
14	2.94×10^{-1}	3.33×10^{-1}
15	2.71×10^{-1}	3.08×10^{-1}
16	2.51×10^{-1}	2.84×10^{-1}
17	2.31×10^{-1}	2.62×10^{-1}
18	2.14×10^{-1}	2.42×10^{-1}
19	1.98×10^{-1}	2.24×10^{-1}
20	1.83×10^{-1}	2.07×10^{-1}
21	1.69×10^{-1}	1.91×10^{-1}
22	1.57×10^{-1}	1.76×10^{-1}
23	1.46×10^{-1}	1.63×10^{-1}
24	1.35×10^{-1}	1.50×10^{-1}
25	1.26×10^{-1}	1.38×10^{-1}
26	1.14×10^{-1}	1.29×10^{-1}
28	6.61×10^{-2}	1.34×10^{-1}
30	-1.02×10^{-2}	1.02×10^{-1}
32	-2.17×10^{-4}	1.31×10^{-3}
33	-5.28×10^{-7}	3.08×10^{-6}
34	-3.03×10^{-12}	1.70×10^{-11}
35	-5.46×10^{-17}	6.14×10^{-17}

Taking the square root of the both sides of (25), it is not difficult to obtain the following incremental iterative formula about parameter t,

$$\begin{aligned} &\Delta t \\ &= \pm\left(\frac{2(\langle \boldsymbol{c}', \boldsymbol{c}' \rangle \langle \boldsymbol{q} - \boldsymbol{c}(t_0), \boldsymbol{q} - \boldsymbol{c}(t_0) \rangle - \langle \boldsymbol{c}', \boldsymbol{q} - \boldsymbol{c}(t_0) \rangle^2)}{k \|\boldsymbol{c}'\|^3}\right)^{\frac{1}{2}}. \end{aligned} \quad (26)$$

In [25], the sign of the incremental iterative formula (26) about parameter t is determined by the sign of right-hand side of (27). Namely, if the inner product $\langle \boldsymbol{c}', \boldsymbol{q} - \boldsymbol{c}(t_0) \rangle$ is positive, then the left-hand side of the incremental iterative formula (26) is positive; otherwise it is negative.

$$sign(\Delta t) = sign(\langle \boldsymbol{c}', \boldsymbol{q} - \boldsymbol{c}(t_0) \rangle). \quad (27)$$

Table 10. Parametric Incremental Iterative Procedure According to $(\Delta u, \Delta v)$ of the GSA with the Test Point $\boldsymbol{p} = (3, 4, 5)$ and the Initial Parametric Value $(2, -2)$ in Example 2

Step	Δu	Δv
1	-1.38×10^{-1}	5.03×10^{-1}
2	-1.25×10^{-1}	4.04×10^{-1}
3	-7.57×10^{-2}	3.82×10^{-1}
4	-2.90×10^{-2}	4.21×10^{-1}
5	-3.08×10^{-2}	1.00×10^{0}
6	-1.05×10^{-2}	-1.46×10^{-1}
7	3.18×10^{-4}	-3.22×10^{-2}
8	-3.81×10^{-5}	-1.02×10^{-3}
9	-9.29×10^{-6}	5.42×10^{-6}
10	-9.78×10^{-7}	6.96×10^{-7}
11	-1.05×10^{-7}	6.84×10^{-8}
12	-1.12×10^{-8}	7.61×10^{-9}
13	-1.20×10^{-9}	7.98×10^{-10}
14	-1.28×10^{-10}	8.61×10^{-11}
15	-1.37×10^{-11}	9.16×10^{-12}
16	-1.46×10^{-12}	9.81×10^{-13}
17	-1.57×10^{-13}	1.05×10^{-13}
18	-1.68×10^{-14}	1.13×10^{-14}
19	-1.62×10^{-15}	1.09×10^{-15}
20	-2.44×10^{-16}	6.83×10^{-17}
21	0	0

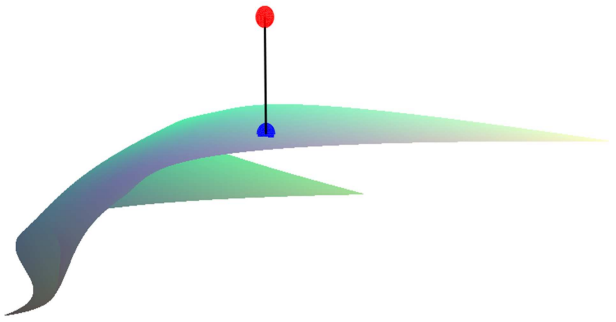


Fig.4. Graphic demonstration for example 2.

In [25], in order to successfully realize the iterative formula of parameter increment Δt, the authors took into account a plane section of the given surface with the initial iterative point $\boldsymbol{p}_0$ being on the plane. According to the expression in [25], there is a parametric curve $\boldsymbol{c}(t)$ with the same parameter where they assumed $\boldsymbol{c}(0) = \boldsymbol{p}_0$. And the tangent vector of the parameter curve $\boldsymbol{c}(t)$ can be expressed by (28),

$$\boldsymbol{c}'(0) = \lambda^1 \boldsymbol{s}_{\cdot 1} + \lambda^2 \boldsymbol{s}_{\cdot 2}. \tag{28}$$

This tangent vector (28) is exactly the tangent vector $\boldsymbol{T} = \alpha_1 \frac{\partial \boldsymbol{s}}{\partial u} + \alpha_2 \frac{\partial \boldsymbol{s}}{\partial v}$ of our paper, where $\lambda_i = \alpha_i, i = 1, 2,$ because point $\boldsymbol{c}(t_0) = \boldsymbol{c}(0)$ is the initial point $\boldsymbol{p}_0$. The radius of the curvature circle of the parameter curve $\boldsymbol{c}(t)$ is defined as $|1/k_n|$, where the curvature circle is located on the plane mentioned above and has its center $\boldsymbol{m} = \boldsymbol{p}_0 + \boldsymbol{n}/k_n$. The test point $\boldsymbol{p}$ is projected onto the curvature circle, and then the curvature circle intersects the line segment $\overline{\boldsymbol{mp}}$ at the intersection point $\boldsymbol{q}$. Because of existing digital errors in the actual operation of computer systems, sometimes the planar curvature circle does not intersect the line segment $\overline{\boldsymbol{mp}}$. To avoid the non-intersect situation happening, we have optimized this case. Under the condition that the center and the radius of the planar curvature circle remain unchanged, we change the planar curvature circle into a spatial curvature sphere. In this way, the spatial curvature sphere and the line segment $\overline{\boldsymbol{mp}}$ can ensure intersection at point $\boldsymbol{q}$ at any time. Δt is calculated through (26) and (27) and $\boldsymbol{c}(\Delta t)$ approximates the orthogonal projection of the test point $\boldsymbol{p}$. Combining (26) and (27), (26) can be naturally transformed into the following formula,

$$\Delta t = \operatorname{sign}(\langle \boldsymbol{T}, \boldsymbol{q} - \boldsymbol{p}_0 \rangle) \left(\frac{2(\|\boldsymbol{T}\|^2 \|\boldsymbol{q} - \boldsymbol{p}_0\|^2 - \langle \boldsymbol{T}, \boldsymbol{q} - \boldsymbol{p}_0 \rangle^2)}{k \|\boldsymbol{c}'\|^3} \right)^{\frac{1}{2}}. \tag{29}$$

Therefore the iterative formula of parameter increments is as follows,

$$\begin{cases} \Delta u = \alpha_1 \Delta t, \\ \Delta v = \alpha_2 \Delta t. \end{cases} \tag{30}$$

After a series of rigorous deduction and analysis, the biggest difference between GSA and the method in [25] is the iterative formula (20) and the iterative formulas (29) and (30), where previous steps of iteration for each of two methods are exactly the same. From the iterative formula (20), it is not difficult to find that it needs 17 operations including calculating five formulas E, F, G, C_4, C_5 and 12 operations of subtracting, multiplying, quotient and squaring while the iterative formulas (29) and (30) require 22 operations through computing of four formulas $\boldsymbol{T}$, $\boldsymbol{q}$, $\boldsymbol{p}_0$, k, subtracting, multiplying, quotient, square root operation, square operation, cubic operation, operation of norm $\|\boldsymbol{T}\|$, $\|\boldsymbol{q} - \boldsymbol{p}_0\|$, inner product $\langle \boldsymbol{T}, \boldsymbol{q} - \boldsymbol{p}_0 \rangle$ and the judgment of positive and negative signs.

Remark 5. GSA in the iterative formula (20) is an orthogonal projection which projects a test point onto a parametric surface $\boldsymbol{s}(u, v)$. For the multiple orthogonal projection points situation, the basic idea of our approach is as follows.

1) Divide a parametric region $[a,b]\times[c,d]$ of parametric surface $\boldsymbol{s}(u,v)$ into M^2 sub-regions $[a_i,a_{i+1}]\times[c_j,c_{j+1}], i,j=0,1,2,...,M-1$, where $a=a_0$, $a_{i+1}-a_i=\frac{b-a}{M}$, $b=a_M$, $c=c_0$, $c_{j+1}-c_j=\frac{d-c}{M}$, $d=c_M$.

2) Randomly select an initial iterative parametric value in each sub-region.

3) Use GSA and use each initial iterative parametric value, do iteration, respectively. Let us assume that the final iterative parametric values are $(\alpha_i,\beta_j), i,j=0,1,2,...,M-1$, respectively.

4) Compute the local minimum distances $d_{ij}, i,j=0,1,2,...,M-1$, where $d_{ij}=\|\boldsymbol{p}-\boldsymbol{s}(\alpha_i,\beta_j)\|$.

5) Compute global minimum distance $d=\|\boldsymbol{p}-\boldsymbol{s}(\alpha,\beta)\|=\min\{d_{ij}\}, i,j=0,1,2,...,M-1$. If we try to find all solutions as soon as possible, divide a parametric region $[a,b]\times[c,d]$ of parametric surface $\boldsymbol{s}(u,v)$ into M^2 sub-regions $[a_i,a_{i+1}]\times[c_j,c_{j+1}], i,j=0,1,2,...,M-1$, where $a=a_0$, $a_{i+1}-a_i=\frac{b-a}{M}$, $b=a_M$, $c=c_0$, $c_{j+1}-c_j=\frac{d-c}{M}$, $d=c_M$ such that M is very large.

Remark 6. In addition to the two examples, we have also tested many other examples. According to these test results, for different initial iterative values, it can converge to the corresponding orthogonal projection point by using GSA, namely, if the initial iterative value is $(u_0,v_0)\in[a,b]\times[c,d]$, which belongs to the parametric region of parametric surface $\boldsymbol{s}(u,v)$, and the corresponding orthogonal projection parametric value for the corresponding orthogonal projection point of the test point $\boldsymbol{p}=(p_1,p_2,p_3)$ is (α,β), then the test point $\boldsymbol{p}$ and its corresponding orthogonal projection parametric value (α,β) satisfy two inequality relationships:

$$\begin{cases}|\langle\boldsymbol{p}-\boldsymbol{s}(\alpha,\beta),\dfrac{\partial\boldsymbol{s}(u,v)}{\partial u}|_{(\alpha,\ \beta)}\rangle|<1\mathrm{E}-14,\\ |\langle\boldsymbol{p}-\boldsymbol{s}(\alpha,\beta),\dfrac{\partial\boldsymbol{s}(u,v)}{\partial v}|_{(\alpha,\ \beta)}\rangle|<1\mathrm{E}-14.\end{cases}$$

According to GSA, these two inequality relationships satisfy the requirement of (31),

$$\begin{cases}|\langle\boldsymbol{p}-\boldsymbol{s}(\alpha,\beta),\dfrac{\partial\boldsymbol{s}(u,v)}{\partial u}|_{(\alpha,\ \beta)}\rangle|=0,\\ |\langle\boldsymbol{p}-\boldsymbol{s}(\alpha,\beta),\dfrac{\partial\boldsymbol{s}(u,v)}{\partial v}|_{(\alpha,\ \beta)}\rangle|=0.\end{cases}\tag{31}$$

Thus it illustrates that GSA is independent of the initial iterative value and GSA is robust and efficient which are satisfied with the previous three of 10 challenges proposed in [36].

5 Conclusions

This paper investigated the problem related to a point projection onto a parametric surface by using normal curvature information. The method is independent of the initial iterative value. Experimental results showed that GSA under consideration is robust and efficient. An area for future research is to develop a more efficient algorithm with higher order convergence for computing the minimum distance between a point and a parametric surface.

Acknowledgements We take the opportunity to thank anonymous reviewers for their thoughtful and meaningful comments, and thank JCST editors for their careful guidance and great help in our paper.

References

[1] Ma Y L, Hewitt W T. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Computer Aided Geometric Design*, 2003, 20(2): 79-99.

[2] Yang H P, Wang W P, Sun J G. Control point adjustment for B-spline curve approximation. *Computer-Aided Design*, 2004, 36(7): 639-652.

[3] Johnson D E, Cohen E. A framework for efficient minimum distance computations. In *Proc. the 1998 IEEE International Conference on Robotics & Automation*, May 1998, pp.3678-3684.

[4] Piegl L, Tiller W. Parametrization for surface fitting in reverse engineering. *Computer-Aided Design*, 2001, 33(8): 593-603.

[5] Pegna J, Wolter F E. Surface curve design by orthogonal projection of space curves onto free-form surfaces. *Journal of Mechanical Design*, 1996, 118: 45-52.

[6] Besl P J, McKay N D. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992, 14(2): 239-256.

[7] Mortenson M E. Geometric Modeling (1st edition). Wiley, 1985.

[8] Zhou J M, Sherbrooke E C, Patrikalakis N. Computation of stationary points of distance functions. *Engineering with Computers*, 1993, 9(4): 231-246.

[9] Limaien A, Trochu F. Geometric algorithms for the intersection of curves and surfaces. *Computers & Graphics*, 1995, 19(3): 391-403.

[10] Polak E, Royset J O. Algorithms with adaptive smoothing for finite minimax problems. *Journal of Optimization: Theory and Applications*, 2003, 119(3): 459-484.

[11] Patrikalakis N, Maekawa T. Shape Interrogation for Computer Aided Design and Manufacturing (1st edition). Springer, 2002.

[12] Johnson D E, Cohen E. Distance extrema for spline models using tangent cones. In *Proc. the 2005 Conference on Graphics Interface*, May 2005, pp.169-175.

[13] Selimovic I. Improved algorithms for the projection of points on NURBS curves and surfaces. *Computer Aided Geometric Design*, 2006, 23(5): 439-445.

[14] Cohen E, Lyche T, Riesebfeld R. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 1980, 14(2): 87-111.

[15] Piegl L, Tiller W. The NURBS Book. Springer, 1995.

[16] Elber G, Kim M S. Geometric constraint solver using multivariate rational spline functions. In *Proc. the 6th ACM Symposiumon Solid Modeling and Applications*, June 2001, pp.1-10.

[17] Park C H, Elber G, Kim K J, Kim G Y, Seong J K. A hybrid parallel solver for systems of multivariate polynomials using CPUs and GPUs. *Computer-Aided Design*, 2011, 43(11): 1360-1369.

[18] Bartoň M. Solving polynomial systems using no-root elimination blending schemes. *Computer-Aided Design*, 2011, 43(12): 1870-1878.

[19] van Sosin B, Elber G. Solving piecewise polynomial constraint systems with decomposition and a subdivision-based solver. *Computer-Aided Design*, 2017, 90: 37-47.

[20] Bartoň M, Elber G, Hanniel I. Topologically guaranteed univariate solutions of underconstrained polynomial systems via no-loop and single-component tests. *Computer-Aided Design*, 2011, 43(8): 1035-1044.

[21] Chen X D, Yong J H, Wang G Z, Paul J C, Xu G. Computing the minimum distance between a point and a NURBS curve. *Computer-Aided Design*, 2008, 40(10/11): 1051-1054.

[22] Chen X D, Xu G, Yong J H, Wang G Z, Paul J C. Computing the minimum distance between a point and a clamped B-spline surface. *Graphical Models*, 2009, 71(3): 107-112.

[23] Oh Y T, Kim Y J, Lee J, Kim M S, Elber G. Efficient point-projection to freeform curves and surfaces. *Computer Aided Geometric Design*, 2012, 29(5): 242-254.

[24] Liu X M, Yang L, Yong J H, Gu H J, Sun J G. A torus patch approximation approach for point projection on surfaces. *Computer Aided Geometric Design*, 2009, 26(5): 593-598.

[25] Hu S M, Wallner J. A second-order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design*, 2005, 22: 251-260.

[26] Li X, Wu Z, Hou L, Wang L, Yue C, Xin Q. A geometric orthogonal projection strategy for computing the minimum distance between a point and a spatial parametric curve. *Algorithms*, 2016, 9(1): Article No. 15.

[27] Li X, Wang L, Wu Z, Hou L, Liang J, Li Q. Convergence analysis on a second-order algorithm for orthogonal projection onto curves. *Symmetry*, 2017, 9(10): Article No. 210.

[28] Hartmann E. On the curvature of curves and surfaces defined by normalforms. *Computer Aided Geometric Design*, 1999, 16(5): 355-376.

[29] Hoschek J, Lasser D. Fundamentals of Computer Aided Geometric Design (1st edition). A K Peters/CRC Press, 1996.

[30] Hu S M, Sun J G, Jin T G, Wang G Z. Computing the parameter of points on Nurbs curves and surfaces via moving affine frame method. *J. Software*, 2000, 11(1): 49-53. (in Chinese).

[31] Liang J, Hou L, Li X, Pan F, Cheng T, Wang L. Hybrid second-order method for orthogonal projection onto parametric curve in n-dimensional Euclidean space. *Mathematics*, 2018, 6(12): Article No. 306.

[32] Li X, Wang L, Wu Z, Hou L, Liang J, Li Q. Hybrid second-order iterative algorithm for orthogonal projection onto a parametric surface. *Symmetry*, 2017, 9(8): Article No. 146.

[33] Li X, Pan F, Cheng T, Wu Z, Liang J, Hou L. Integrated hybrid second order algorithm for orthogonal projection onto a planar implicit curve. *Symmetry*, 2018, 10(5): Article No. 164.

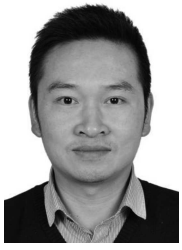
[34] Ko K H, Sakkalis T. Orthogonal projection of points in CAD/CAM applications: An overview. *Journal of Computational Design and Engineering*, 2014, 1(2): 116-127.

[35] Wang X P, Zhang W Z, Huang X. Computation of point inversion and ray-surface intersection through tracing along the base surface. *The Visual Computer*, 2015, 31(11): 1487-1500.

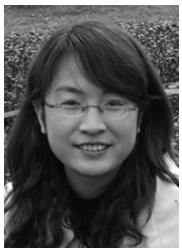
[36] Piegl L A. Ten challenges in computer-aided design. *Computer-Aided Design*, 2005, 37(4): 461-470.

Xiaowu Li received his M.S. degree in computer science from Chongqing University, Chongqing, in 2006, and Ph.D. degree in mathematics science from Chongqing University, Chongqing, in 2011. He is currently a professor of College of Data Science and Information Engineering in Guizhou Minzu University, Guiyang. His research interests include numerical solution of partial differential equation, pattern recognition, computation geometry and computer aided geometric design.

Zhinan Wu received his M.S. degree in computer science from Chongqing University, Chongqing, in 2006. He is currently a doctoral candidate at Jiangxi University of Finance and Economics, Nanchang, and also currently an associate professor in Yichun University, Yichun. His research interests include software engineering, pattern recognition, image processing, computation geometry and computer aided geometric design.



Feng Pan received his M.S. degree in computer software and theory from Guizhou University, Guiyang, in 2008. He is currently a doctoral candidate of South China University of Technology, Guangzhou. He is a senior member of CCF. His research interests include software engineering, big data and machine learning.



Juan Liang received her M.S. degree in applied mathematics from Chongqing University, Chongqing, 2013, and her B.S. degree in applied mathematics from Datong University, Datong, 2010. She works at Taiyuan Institute of Technology, Taiyuan, currently. Her research interests include complex system modeling and simulation, statistical analysis and data processing.

Jiafeng Zhang received his Ph.D. degree in computer science and technology from Southwest Jiaotong University, Chengdu, in 2014, his M.S. and B.S. degrees in applied mathematics from Southwest Jiaotong University, Chengdu, and Huaibei Normal University, Huaibei, in 2007 and 2004, respectively. He is currently a professor at the School of Data Science and Information Engineering in Guizhou Minzu University, Guiyang. His main research interests include automated reasoning and multiple-valued logic.

Linke Hou received his Ph.D. degree in agricultural and applied economics from Chinese Academy of Sciences, Beijing, in 2012. He is currently an associate professor at the Center for Economic Research, Shandong University, Jinan. His main research interests include computation in economics and social networks.