Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 3790762, 12 pages
https://doi.org/10.1155/2019/3790762

Research Article

An Approach to Computing Multipoint Inversion and Multiray Surface Intersection on Parametric Surface

Pei Jingyu, Wang Xiaoping, and Zhang Leen

College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China

Correspondence should be addressed to Pei Jingyu; jypei@nuaa.edu.cn

Received 7 March 2019; Revised 22 April 2019; Accepted 19 May 2019; Published 11 June 2019

Academic Editor: Nhon Nguyen-Thanh

This article presents a method for multipoint inversion and multiray surface intersection problem on the parametric surface. By combining tracing along the surface and classical Newton iteration, it can solve point inversion and ray-surface intersection issues concerning a large number of points or rays in a stable and high-speed way. What is more, the computation result can approximate to exact solutions with arbitrary precision because of the self-correction of Newton-Raphson iteration. The main ideas are adopting a scheme tracing along the surface to obtain a good initial point, which is close to the desired point with any prescribed precision, and conducting Newton iteration process with the point as a start point to compute desired parameters. The new method enhances greatly iterative convergence rate compared with traditional Newton's iteration related methods. In addition, it has a better performance than traditional methods, especially in dealing with multipoint inversion and multiray surface intersection problems. The result shows that the new method is superior to them in both speed and stability and can be widely applied to industrial and research field related to CAD and CG.

1. Introduction

Point inversion and ray-surface intersection are significant problems, which are to compute the parametric coordinates value of a point by its Cartesian coordinates or ray tangent vectors. Many computational geometry problems, computer vision problems, and computer graphics applications are closely related to surface manipulation. For a parametric surface, the most ordinary manipulation might be to find the parameters of points on the surface in a fast and stable way. Several efforts have been made towards point inversion issue of polynomial parametric surface [1–5]. For approximate methods, Hoschek et al. [6] and Hartmann et al. [7] proposed the first-order geometric iteration method. Hu and Wallner [8] gave out a second-order iteration by computing a normal curvature circle and a line segment intersection point. Liu et al. [9] enhanced the stability and speeded up the convergence by changing the normal curvature circle into a torus patch. Some classic methods are provided by Piegl and Tiller [10] by decomposing a NURBS surface into quadrilaterals, projecting the test point onto the closest quadrilateral, and then recovering the parameter from the closest quadrilateral. Ma and Hewitt [11] put forward an algorithm to obtain a good initial value for the Newton-Raphson method. Xu [5] developed an applicable method by subdividing the parametric domain and checking the signs of all coefficients of a Bernstein polynomial. For NURBS curve and surface, Johnson and Cohen [12] concentrated on the polygonal model for finding the minimum distance between two geometric objects. Chin [13] and Edelsbruner [14] found an algorithm with complexity $O(\log N)$ for finding the minimum distance between two convex polygons. For ray-surface intersection, Taezoon et al. [15] provided a second order geometric iteration which provides better approximation to the local shape of the surface. Wang et al. [16] provided a method combining Bezier clipping and Newton's method. Wang et al. [17] presented an algorithm for enhancing the performance of both numerical and subdivision methods. Saini [18] gave an application for inverse geometric reconstruction of conics in 3D space by using a ray-surface intersection. For parametric model, Mortensen [19] gave a numerical approach for this problem through finding a vector from the point on the

curve that is perpendicular to the tangent vector. Song et al. [20] constructed a biarc that locally approximates a segment on the original curve starting from the current projective point. Ko [21] reviewed methods for computing orthogonal projection of points onto curves and surfaces, which are given on implicit or parametric form or as points cloud. Through parameterizing the progenitor curve, a novel curve projection scheme is proposed by Xu [22] for computing the orthogonal projection. Limaiem [23] presented another approach to find the minimum distance to convex parametric curves and surfaces. Lin [24] provided the approach for finding the minimum distance for concave surfaces. For further reading, papers [25, 26] presented several algorithms for computing the minimum distance between a point and a surface. With the Newton-Raphson iteration initial value Piegl and Tiller [10] provided another method to solve the point projection problem for NURBS surface.

Wang et al. [27] developed several methods for point inversion and ray surface intersection issues by constructing and projecting a line segment and tracing the parameters along the projected curve or intersection curve from an arbitrarily prescribed initial point. To some extent, the approach avoids fully and truly the sensitivity to the choice of the initial value in those Newton-Raphson iteration related methods. Young-Taek Oh et al. [28] used an efficient culling technique to reduce redundant curves and surfaces and proposed an efficient method to project a point to its nearest point on a set of freedom surfaces and curves. Chen et al. [29] revised a rational cubic clipping approach to get two bounding cubes within $O(n)$ time, which could get a faster convergence rate.

So far, these existing methods are good at solving point inversion and ray-surface intersection problems only of a single point or several points. However, many engineering issues such as composite material fibre placement and nonplane optical imaging often need to deal with point inversion or ray-surface intersection issues of a large number points simultaneously. Also it needs high computational efficiency (high precision and high success rate). There are two demerits in using traditional Newton related methods to handle the multipoint inversion and multiray surface intersection problems. One is totally time-consumption in finding initial value usually through subdivision for every point. Another is the sensitivity of Newton's method to an initial guess. As a matter of fact, when sensitivity occurs the iteration process fails to converge. Thus user must restart the whole computational process. So the two disadvantages lead to huge computational costs. In practice, even 5% failure rates of iteration will result in the failure of a path planning process for robotic fibre placement. Therefore we need to develop new method towards the multipoint inversion problems.

Enlightened by the method [27], which can trace solution with any prescribed initial point and Newton iteration's fast computation, we try to resolve the new problem by combining the two existing methods and improving them. Our main contributions are as follows: overcome low efficiency of traditional methods when dealing with multipoint inversion issue; reduce and even eliminate the sensitivity of traditional Newton's methods to an initial guess; propose a new algorithm good at multipoint inversion and multiray surface intersection problems.

2. Fundamentals of Algorithm

Here algorithm foundations are introduced and deduced, which are the indispensable parts of the method. Actually the new algorithm of multipoints inversion and multiray surface intersection can be divided into two steps. First, we improve the geometric method to obtain a good initial value. The second important step is to enhance the accuracy of solving the differential equations through the classic Newton-Raphson iteration.

2.1. Multipoint Inversion. Let us consider a parametric surface $\boldsymbol{S}(u, v), (u, v) \in D \subset \mathbb{R}^2$ (a mapping from D $\longrightarrow \mathbb{R}^3$) with at least 2-order derivative existing. For any $(u, v) \in D$ there are $\boldsymbol{S}_u \neq \boldsymbol{0}$ and $\boldsymbol{S}_v \neq \boldsymbol{0}$. The problem is as follows: $\boldsymbol{P}$ is a point on the surface $\boldsymbol{S}$, with its Cartesian coordinates known. See Figure 1. We want to compute its parametric coordinates $(u, v) \in D$. Firstly take a point $\boldsymbol{P}_0(u_0, v_0)$ arbitrarily on the given surface $\boldsymbol{S}$. Construct a line segment $\boldsymbol{L}(t)$ with $\boldsymbol{P}$ and $\boldsymbol{P}_0$ as follows:

$$\boldsymbol{L}(t) = t\boldsymbol{P} + (1 - t)\boldsymbol{P}_0, \quad 0 \leq t \leq 1 \tag{1}$$

Calculate a normal vector $\boldsymbol{N}_0$ of the known point $\boldsymbol{P}_0$ on the surface $\boldsymbol{S}$. Then define a vector $\boldsymbol{V}$ as $\boldsymbol{V} = [\boldsymbol{N}_0 \times (\boldsymbol{P} - \boldsymbol{P}_0)] \times (\boldsymbol{P} - \boldsymbol{P}_0)$, which represents the project direction of line segment $\boldsymbol{L}(t)$. Let the projected curve of $\boldsymbol{L}(t)$ onto the surface $\boldsymbol{S}$ be $\boldsymbol{P}(t)$, which can be characterized as follows

$$[\boldsymbol{P}(t) - \boldsymbol{L}(t)] \times \boldsymbol{V} = 0. \tag{2}$$

Computing the derivative on both sides of Equation (2) with respect to t, it follows that

$$[\boldsymbol{S}_u \times \boldsymbol{V} \quad \boldsymbol{S}_v \times \boldsymbol{V}] \begin{bmatrix} u_t \\ v_t \end{bmatrix} = \boldsymbol{L}_t \times \boldsymbol{V}. \tag{3}$$

Taking the dot product of both sides of (3) with the vector $\boldsymbol{S}_u$ and $\boldsymbol{S}_v$ separately yields

$$\begin{bmatrix} \boldsymbol{S}_u \cdot (\boldsymbol{S}_u \times \boldsymbol{V}) & \boldsymbol{S}_u \cdot (\boldsymbol{S}_v \times \boldsymbol{V}) \\ \boldsymbol{S}_v \cdot (\boldsymbol{S}_u \times \boldsymbol{V}) & \boldsymbol{S}_v \cdot (\boldsymbol{S}_v \times \boldsymbol{V}) \end{bmatrix} \begin{bmatrix} u_t \\ v_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_u \cdot [(\boldsymbol{P} - \boldsymbol{P}_0) \times \boldsymbol{V}] \\ \boldsymbol{S}_v \cdot [(\boldsymbol{P} - \boldsymbol{P}_0) \times \boldsymbol{V}] \end{bmatrix}. \tag{4}$$

Thus, if the mixed product $[\boldsymbol{V}, \boldsymbol{S}_u, \boldsymbol{S}_v]$ is not equal to 0, by solving the matrix equation (4), we have u_t and v_t as follows:

$$\begin{bmatrix} u_t \\ v_t \end{bmatrix} = \begin{bmatrix} 0 & \dfrac{1}{\boldsymbol{S}_v \cdot (\boldsymbol{S}_u \times \boldsymbol{V})} \\ \dfrac{1}{\boldsymbol{S}_u \cdot (\boldsymbol{S}_v \times \boldsymbol{V})} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_u \cdot [(\boldsymbol{P} - \boldsymbol{P}_0) \times \boldsymbol{V}] \\ \boldsymbol{S}_v \cdot [(\boldsymbol{P} - \boldsymbol{P}_0) \times \boldsymbol{V}] \end{bmatrix} = \begin{bmatrix} \dfrac{\boldsymbol{S}_v \cdot [\boldsymbol{V} \times (\boldsymbol{P} - \boldsymbol{P}_0)]}{\boldsymbol{V} \cdot (\boldsymbol{S}_u \times \boldsymbol{S}_v)} \\ \dfrac{\boldsymbol{S}_u \cdot [(\boldsymbol{P} - \boldsymbol{P}_0) \times \boldsymbol{V}]}{\boldsymbol{V} \cdot (\boldsymbol{S}_u \times \boldsymbol{S}_v)} \end{bmatrix}. \tag{5}$$

$$\left| \boldsymbol{P} - \boldsymbol{S}(u_n, v_n) \right| < \epsilon_1. \tag{6}$$

Equation (5) is the govern equation of multipoint inversion. To solve the 1st order matrix differential equation (5), we will obtain initial value for the Newton-Raphson method. Inequality (6) is the convergence condition.

2.2. Multiray Surface Tracing. The computing ideas can also apply to the multiray surface tracing, which provides the basic algorithm for ray-surface intersection. Traditional methods for ray-surface intersection problem can be roughly classified into Newton-iteration-based ones and subdivision-based ones. Here we give a new method to solve this problem. See Figure 2. Similarly to point inversion, we reduce the problem to the projection of straight line segment onto the related surface.

The ray line can be expressed as

$$\boldsymbol{R}(t) = \boldsymbol{O} + \boldsymbol{k}t, \quad t \geq 0 \tag{7}$$

Firstly, choose an initial point $\boldsymbol{P}_0(u_0, v_0)$ on the base surface $\boldsymbol{S}$, calculate the foot point $\boldsymbol{P}_j$ projects from $\boldsymbol{P_0}$ to $\boldsymbol{R}(t)$ as follows:

$$\boldsymbol{P}_j = \boldsymbol{O} + \frac{[(\boldsymbol{P}_0 - \boldsymbol{O}) \cdot \boldsymbol{k}] \cdot \boldsymbol{k}}{\boldsymbol{k} \cdot \boldsymbol{k}} \tag{8}$$

Construct a line segment $\boldsymbol{L}(t)$ as follows:

$$\boldsymbol{L}(t) = (1 - t)\boldsymbol{P}_0 + t\boldsymbol{P_j}, \quad 0 \leq t \leq 1 \tag{9}$$

Then we have the following characteristic equation:

$$[\boldsymbol{L}(t) - \boldsymbol{O}] \times [\boldsymbol{P}(t) - \boldsymbol{O}] = \boldsymbol{0} \tag{10}$$

Taking the derivative of (10) with respect to t, we get

$$\begin{aligned} &\boldsymbol{S}_u \times [\boldsymbol{L}(t) - \boldsymbol{O}] \frac{du}{dt} + \boldsymbol{S}_v \times [\boldsymbol{L}(t) - \boldsymbol{O}] \frac{dv}{dt} \\ &\quad = \boldsymbol{L}'(t) \times [\boldsymbol{P}(t) - \boldsymbol{O}] \end{aligned} \tag{11}$$

Assume

$$(\boldsymbol{S}_u \times \boldsymbol{S}_v) \cdot [\boldsymbol{L}(t) - \boldsymbol{O}] \neq \boldsymbol{0} \tag{12}$$

along the projective curve segment. Finally we have

$$\begin{aligned} \frac{du}{dt} &= \frac{\left[(\boldsymbol{S}(u, v) - \boldsymbol{O}) \times \left(\boldsymbol{P_j} - \boldsymbol{P_0}\right)\right] \cdot \boldsymbol{S}_v}{(\boldsymbol{S}_u \times \boldsymbol{S}_v) \cdot [\boldsymbol{L}(t) - \boldsymbol{O}]} \\ \frac{dv}{dt} &= \frac{\left[\left(\boldsymbol{P_j} - \boldsymbol{P_0}\right) \times (\boldsymbol{S}(u, v) - \boldsymbol{O})\right] \cdot \boldsymbol{S}_u}{(\boldsymbol{S}_u \times \boldsymbol{S}_v) \cdot [\boldsymbol{L}(t) - \boldsymbol{O}]} \end{aligned} \tag{13}$$

Adding initial value conditions

$$\begin{aligned} u(0) &= u_0, \\ v(0) &= v_0 \end{aligned} \tag{14}$$

By solving (13), we can obtain the parameters of point $\boldsymbol{P}$.

2.3. Improvement on the ODE Solver. ODE45 is a function implementing a Runge-Kutta method with a variable time step for efficient computation. ODE45 is designed to handle the following general problem:

$$\frac{dx}{dt} = \boldsymbol{f}(t, \boldsymbol{x}), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{15}$$

where t is the independent variable, $\boldsymbol{x}$ is a vector of dependent variables to be found and $\boldsymbol{f}(t, \boldsymbol{x})$ is a function of t and $\boldsymbol{x}$. The mathematical problem is specified when $\boldsymbol{f}(t, \boldsymbol{x})$ on the right-hand side of (15) is set and the initial conditions $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ at time t_0 are given.

The disadvantage of using ODE45 is slow convergence because of the variable time step. When the tracing point $\boldsymbol{P}_n$ is very close to the $\boldsymbol{P}$, ODE45 has to change the step size very frequently. It takes plenty of time to satisfy convergence conditions. In order to overcome the disadvantages above, new method will use fixed step 4th order Runge-Kutta method to trace along the base surface. Equations (5) and (13) can be converted as follows:

$$\boldsymbol{u}_t = \boldsymbol{F}(t, \boldsymbol{u}) \tag{16}$$

where $\boldsymbol{u}_t = \begin{bmatrix} u_t(t) \\ v_t(t) \end{bmatrix}$, $\boldsymbol{u} = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}$.

The explicit 4th-order fixed-step Runge-Kutta calculation formulas are as follows:

$$\begin{aligned} \boldsymbol{u}_{n+1} &= \boldsymbol{u}_n + \frac{\Delta t}{6}(\boldsymbol{K}_1 + 2\boldsymbol{K}_2 + 2\boldsymbol{K}_3 + \boldsymbol{K}_4) \\ \boldsymbol{K_1} &= \boldsymbol{F}(t_n, \boldsymbol{u}_n) \\ \boldsymbol{K}_2 &= \boldsymbol{F}\left(t_n + \frac{\Delta t}{2}, \boldsymbol{u}_n + \frac{\Delta t}{2}\boldsymbol{K}_1\right) \\ \boldsymbol{K}_3 &= \boldsymbol{F}\left(t_n + \frac{\Delta t}{2}, \boldsymbol{u}_n + \frac{\Delta t}{2}\boldsymbol{K}_2\right) \\ \boldsymbol{K}_4 &= \boldsymbol{F}(t_n + \Delta t, \boldsymbol{u}_n + \Delta t\boldsymbol{K}_3) \end{aligned} \tag{17}$$

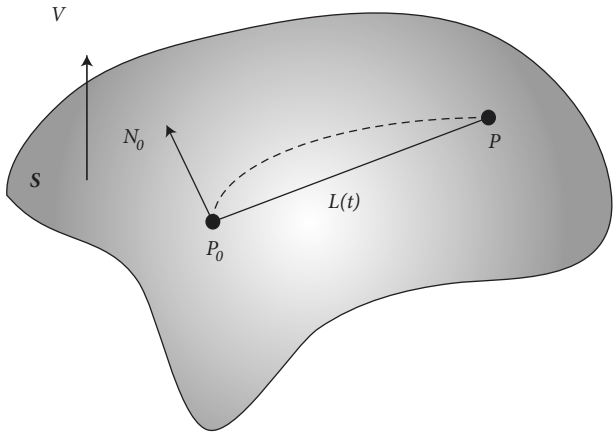

Figure 1: Geometry diagram of projection method.

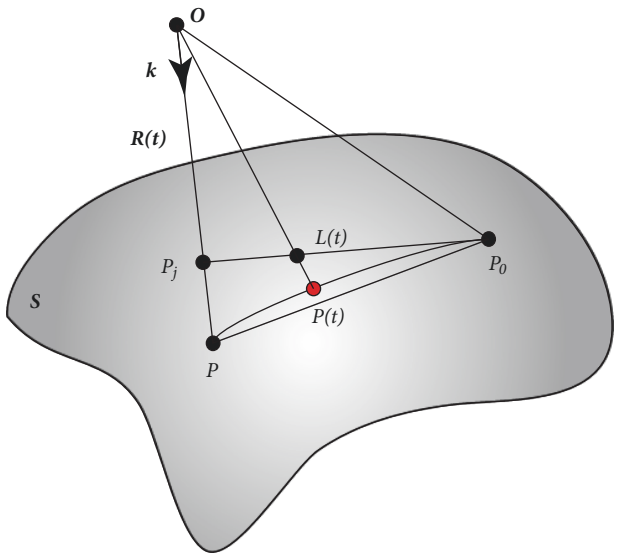

Figure 2: Geometry diagram of multiray surface tracing.

where Δt is fixed step satisfying

$$t_{i+1} - t_i = \Delta t, \quad i = 0, 1, 2, \ldots, n, \tag{18}$$

$$t_0 = 0, \qquad t_n = 1 \tag{19}$$

The end computing conditions for multipoint inversion are inequality (6) and $0 \le t \le 1$.

For multiray surface intersection, when the projection distance d from surface point $\boldsymbol{P}$ to the ray $\boldsymbol{R}(t)$ is small enough, the point $\boldsymbol{P}$ is considered as exact solution. It can be described as follows:

$$d = \frac{|\boldsymbol{OP} \cdot \boldsymbol{k}|}{|\boldsymbol{k}|} \le \epsilon_1 \tag{20}$$

So, the convergence condition for multiray surface intersection algorithm is inequality (20) and $0 \le t \le 1$. The two conditions can guarantee the algorithm not go into an infinite loop.

The precision of the 4^{th}-order Runge-Kutta is $c\Delta t^5$ in every computation step, where c is a positive arbitrary constant. After tracing along the base surface $\boldsymbol{S}$, the accumulated error $\boldsymbol{e}$ will be very great. Paper [30] analysed systematically the accuracy of Runge-Kutta method. For a known surface $\boldsymbol{S}$, the truncation error between exact solution and resulting solution $|E|$ in each calculation step has an upper bound as follows:

$$|E| \le \frac{73}{720} ML^4 h^5 \tag{21}$$

where L and M are positive constants independent of t, $\boldsymbol{u}$, and h.

2.4. Newton-Raphson Iteration. The second-order algorithm of Hu et al. [31] is a good representation of Newton-Raphson method. By calculating the normal curvature circle of base surface at $\boldsymbol{P}_0$, a linear combination of tangent vectors is obtained. Connect the point $\boldsymbol{P}$ with the centre of normal curvature circle $\boldsymbol{Q}$. The intersection point of the line segment $\boldsymbol{PQ}$ and the normal curvature circle is $\boldsymbol{X}$. Project the point $\boldsymbol{X}$ along the normal vector at $\boldsymbol{P}_0$ onto the base surface. The foot of perpendicular from $\boldsymbol{X}$ to the base surface is approximated by a known function $f(\Delta t)$. With Δt computed, a new set of parameters can be obtained for next iteration. However, the problem of the method is the stiffness equations which consist of two tangent vectors and a normal vector. According to the numerical experiment, if the point $\boldsymbol{P}$ is too far from $\boldsymbol{P}_0$ or the curvature of base surface is too small, the coefficients of tangent vectors will be much greater than the coefficients of normal vector. The phenomenon will cause the linear combination equation to have no solution, especially when solving multipoints inversion.

The new method combining the Runge-Kutta algorithm and the Newton-Raphson algorithm can avoid the problem in the second order of Hu et al. [8]. Meanwhile, it can also save lots of time, especially in calculating multipoints inversion. The total CPU time will be economized violently, with the calculation points increasing. For the calculation process, after tracing along the given surface, a result point $\boldsymbol{P}_n$ will be found, which is not close enough to point $\boldsymbol{P}$. The point $\boldsymbol{P}_n$ is taken as the initial value of the Newton-Raphson iteration. Usually, Newton-Raphson iteration needs a good initial value, so that the computation can converge. According to equation (19), selecting a suitable step size of ODE solver can assure the initial value good enough to the Newton-Raphson iteration. Piegl [32] shows a calculation process to solve point inversion by Newton-Raphson iteration. Without a good initial value, the method often causes numerical oscillation. There computation process is listed as follows:

$$\boldsymbol{r}(u, v) = \boldsymbol{S}(u, v) - \boldsymbol{P} \tag{22}$$

and two scalar equations

$$f(u, v) = \boldsymbol{r}(u, v) \cdot \boldsymbol{S}_u(u, v) \tag{23}$$

$$g(u, v) = \boldsymbol{r}(u, v) \cdot \boldsymbol{S}_v(u, v). \tag{24}$$

Equations (23) and (24) can be written as a matrix form

$$\delta_i = \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} u_{i+1} - u_i \\ v_{i+1} - v_i \end{bmatrix} \tag{25}$$

$$\boldsymbol{J}_i = \begin{bmatrix} f_u & f_v \\ g_u & g_v \end{bmatrix} = \begin{bmatrix} |\boldsymbol{S}_u|^2 + \boldsymbol{r} \cdot \boldsymbol{S}_{uu} & \boldsymbol{S}_u \cdot \boldsymbol{S}_v + \boldsymbol{r} \cdot \boldsymbol{S}_{uv} \\ \boldsymbol{S}_u \cdot \boldsymbol{S}_v + \boldsymbol{r} \cdot \boldsymbol{S}_{vu} & |\boldsymbol{S}_v|^2 + \boldsymbol{r} \cdot \boldsymbol{S}_{vv} \end{bmatrix} \tag{26}$$

$$\kappa_i = -\begin{bmatrix} f \\ g \end{bmatrix} \tag{27}$$

where all items in the matrix $\boldsymbol{J}_i$ and κ_i are calculated at (u_i, v_i). At the ith iteration we need to solve a 2×2 system of linear equations in the unknown δ_i given by

$$\boldsymbol{J}_i \delta_i = \kappa_i \tag{28}$$

Convergence criteria are given by

$$\left| \boldsymbol{P} - \boldsymbol{S}(u_i, v_i) \right| \leq \epsilon_2 \tag{29}$$

However, for multiray surface intersection, the computation process can be described as

$$\boldsymbol{G}(u, v, t) = \boldsymbol{S}(u, v) - \boldsymbol{R}(t) = \boldsymbol{0} \tag{30}$$

in which $\boldsymbol{S}(u, v)$ is a parametric surface and $\boldsymbol{R}(t)$ is a ray in 3D space. Given a point $\boldsymbol{P}$ and a ray, a plane containing them can be created as illustrated in Figure 2. Then the intersection of the generated plane and the surface products a curve $\boldsymbol{c}$, which is indicated as a dotted curve in Figure 2. Since the intersection curve is on the plane, it becomes a planar curve and subsequently the intersection point between the ray and the surface should lie on the intersection curve $\boldsymbol{c}$. So, the same method developed for the ray and 2D planar curve intersection case is directly applicable for this problem.

The basic approach to solve (30) is identical to the case of finding intersection between a ray and a planar curve. Given $\boldsymbol{x} = (u, v, t)^T$, the amount of update for $\boldsymbol{x}$ is obtained by solving equation as follows:

$$\Delta \boldsymbol{x} = -\boldsymbol{J}^{-1} \boldsymbol{G}(\boldsymbol{x}) \tag{31}$$

The Jacobian matrix $\boldsymbol{J}$ is given by

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{S}_u & \boldsymbol{S}_v & \boldsymbol{k} \end{bmatrix} \tag{32}$$

In which $\boldsymbol{S}_u$, $\boldsymbol{S}_v$, and $\boldsymbol{k}$ are column vectors.

Consider a point $\boldsymbol{P}$ on the surface. The intersection between the ray and the tangent plane is computed by plugging the ray equation into the tangent plane equations, yielding the value of t as follows:

$$t = \frac{(\boldsymbol{P} - \boldsymbol{O}) \cdot \boldsymbol{N}}{\boldsymbol{k} \cdot \boldsymbol{N}} \tag{33}$$

in which $\boldsymbol{N}$ is the unit surface normal vector of $\boldsymbol{P}$.

Convergence criteria are given by

$$\frac{\left\| (\boldsymbol{S}(u_i, v_i) - \boldsymbol{O}) \times \boldsymbol{k} \right\|}{\|\boldsymbol{k}\|} \leq \epsilon_2 \tag{34}$$

The tolerance ϵ_2 is much smaller than the tolerance ϵ_1. If the inequality (29) or (34) is satisfied, then the calculation is considered successful.

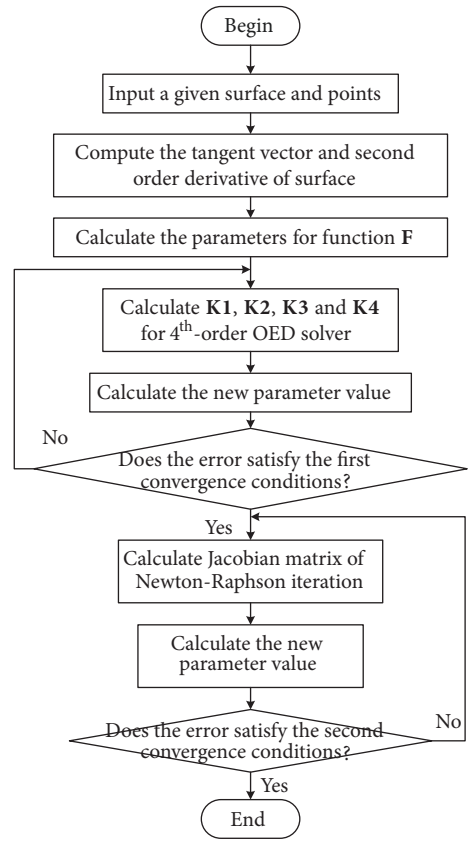

Figure 3: The calculation flowchart.

3. Computational Process of the Proposed New Method

A single point calculation is taken as an example in the computational process to explain the whole process which combines tracing along the base surface and the Newton-Raphson iteration. Figure 3 shows the whole calculation flowchart for the new algorithm.

The main pseudocode of new method with Newton-Raphson iteration are showed in Algorithm 1. When solving multipoints inversion, the default $\boldsymbol{O}$ coincides with $\boldsymbol{P}_0$.

The pseudocodes of the ODE solver are showed in Algorithm 2. The pseudocode of Newton-Raphson iterations are showed in Algorithm 3.

In practical, the algorithm transforms the initial value problem of Newton iteration into the design problem of ODE solver error. By reasonably designing the error of the ODE solver, the Newton iteration solver can obtain an excellent initial value.

4. Convergence Rate Analysis

In this chapter we will analyse the convergence rate. There is a theorem about convergence of Newton iteration. The proof of the theorem can be found in section 10.2.2 of book [33].

Theorem. Assuming $\boldsymbol{F} : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is G-differentiable in an open neighbourhood $S_0 \subset D$ of point $\boldsymbol{F}\boldsymbol{x}^* = \boldsymbol{0}$, $\boldsymbol{x}^* \in D$.

Require: $S(u, v) \bigvee P \bigvee P_0 \bigvee u_0$
1: **procedure** MAIN
2: **if** $\|O - P_0\| \neq 0$ **then**
3: $k \longleftarrow (O - P_0)/\|O - P_0\|$
4: **end if**
5: $t \longleftarrow 0$
6: $h \longleftarrow h_0$
7: $u \longleftarrow ODE(h, u_0, P, P_0, O, k)$
8: $P_n \longleftarrow S(u[1], u[2])$
9: $t \longleftarrow t + h$
10: **if** $\|O - P_0\| = 0$ **then**
11: **while** $\|P_n - P\| > \epsilon_1 \& t \leq 1$ **do**
12: $u_0 \longleftarrow u$
13: $u \longleftarrow ODE(h, u_0, P, P_0, O, k)$
14: $P_n \longleftarrow S(u[1], u[2])$
15: $t \longleftarrow t + h$
16: **end while**
17: **else**
18: **while** $|OP \cdot k|/\|k\| > \epsilon_1 \& t \leq 1$ **do**
19: $u_0 \longleftarrow u$
20: $u \longleftarrow ODE(h, u_0, P, P_0, O, k)$
21: $P_n \longleftarrow S(u[1], u[2])$
22: $t \longleftarrow t + h$
23: **end while**
24: **end if**
25: **if** $\|O - P_0\| = 0$ **then**
26: **while** $\|P_n - P\| > \epsilon_2$ **do**
27: $u_0 \longleftarrow u$
28: $u \longleftarrow NewtonIteration(u_0, P.O, k)$
29: $P_n \longleftarrow S(u[1], u[2])$
30: **end while**
31: **else**
32: **while** $|OP \cdot k|/\|k\| > \epsilon_2$ **do**
33: $u_0 \longleftarrow u$
34: $u \longleftarrow NewtonIteration(u_0, P.O, k)$
35: $\longleftarrow S(u[1], u[2])$
36: **end while**
37: **end if**
38: **end procedure**

ALGORITHM 1: The Main algorithm.

Assuming F' is continuous at x^* and $F'(x^*)$ is not singular. Then x^* is an attraction point of Newton iteration

$$\mathcal{J} : x^{(k+1)} = x^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right), \quad k = 0, 1, \ldots, \tag{35}$$

In addition, if there is a constant $\alpha < +\infty$ and $p \in (0, 1]$ satisfying

$$\left\|F'(x) - F'\left(x^*\right)\right\| \leq \alpha \left\|x - x^*\right\|^p, \quad \forall x \in S_0 \tag{36}$$

then $O_R(\mathcal{J}, x^*) \geq O_Q(\mathcal{J}, x^*) \geq 1 + p$. At last, if F is continuously differentiable on S_0, and its second-order F-derivative exists which satisfies

Require: $S_u \longleftarrow \partial S/\partial u \bigvee S_v \longleftarrow \partial S/\partial v$
1: **function** ODE
2: **for** $i = 0 \longrightarrow 3$ **do**
3: **if** $\|O - P_0\| = 0$ **then**
4: $du_i \longleftarrow Equ(5)(P, P_0, u_t, h)$
5: **else**
6: $du_i \longleftarrow Equ(13)(P, P_0, u_t, h, k)$
7: **end if**
8: **end for**
9: $K_1 \longleftarrow du_0$
10: $u_1 \longleftarrow du_0 + h \times K_1/2$
11: $K_2 \longleftarrow du_1$
12: $u_2 \longleftarrow du_1 + h \times K_2/2$
13: $K_3 \longleftarrow du_2$
14: $u_3 \longleftarrow du_2 + h \times K_3$
15: $K_4 \longleftarrow du_3$
16: $u_3 \longleftarrow du_3$
17: $y \longleftarrow u + (h/6)(K_1 + 2K_2 + 2K_3 + K_4)$ **return** y
18: **end function**

ALGORITHM 2: The ODE Solver.

$$F''\left(x^*\right) hh \neq 0, \quad \forall h \in \mathbb{R}^n, \ h \neq 0 \tag{37}$$

then $O_R(\mathcal{J}, x^*) = O_Q(\mathcal{J}, x^*) = 2$.

The theorem shows that there is always an attraction domain S_0. If the initial approximation is in S_0, the sequence $\{x^{(k)}\}$ generated by Newton-Raphson iteration always lies in S_0 and converges to x^*. Roughly speaking, iterating once with Newton-Raphson iteration method can double the number of significant digits. That means Newton-Raphson iteration converges quickly. It also should be noted that the Newton-Raphson iteration method is self-correcting. That is to say, $x^{(k+1)}$ only depends on F and $x^{(k)}$. So, the rounding error generated by the previous iteration will not be passed down step by step. That is the reason why the combination of Newton-Raphson iteration and Runge-Kutta method can obtain precise result.

In this paper, the Runge-Kutta method and the Newton iteration we provided are both first-order convergent. However, Wang's method needs to build multiline tracing in order to make the error satisfy the convergence condition. That will make the step size of ODE far less than the error ϵ_2. Therefore, Wang's method has a precision limitation when using the fix-step Runge-Kutta method.

Moreover, there are two reasons for the proposed method being faster than Hu's second order method. One is Hu's second-order method needs to subdivide the base surface when calculating each point. That will cost a lot of time. The other one is related to its success rate. According to the experiment, Hu's second-order often brings a failure rate around 6%. That will slow down the algorithm.

For single point calculation, Hu's second order method has a higher convergence order than others. But when calculating large number points on base surface, Hu's second-order method needs to subdivide the base surface in order to obtain a good initial point. Strictly speaking, when calculating a

Require: $S_u \longleftarrow \partial S/\partial u \bigvee S_v \longleftarrow (\partial S/\partial v)S_u \longleftarrow \partial S/\partial u \bigvee S_v \longleftarrow \partial S/\partial v \bigvee S_{uu} \longleftarrow \partial^2 S/\partial u^2 \bigvee S_{vv} \longleftarrow \partial^2 S/\partial v^2 \bigvee S_{uv} \longleftarrow \partial^2 S/\partial u \partial v \bigvee S_{vu} \longleftarrow \partial^2 S/\partial v \partial u$

1: **function** NewtonIteration
2: **if** $\|O - P_0\| = 0$ **then**
3: $J \longleftarrow Equ(26)(P, u_0)$
4: $delta \longleftarrow Equ(25)(P, u_0)$
5: $u \longleftarrow u + delta$
6: **else**
7: $du_i \longleftarrow Equ(32)(k, u_0)$
8: $\delta x \longleftarrow Equ(31)(k, u_0)$
9: $u[1] \longleftarrow u[1] + \delta x[1]$
10: $u[2] \longleftarrow u[2] + \delta x[2]$
11: **end ifreturn** u
12: **end function**

Algorithm 3: The Newton Iteration.

failure point, Hu's second-order method needs to resubdivide the base surface for a better initial point. That is time wasting.

5. Implementation Examples

5.1. Multipoint Inversion Cases. Here we will use practical examples to demonstrate why the proposed method is better than the latest method [27] and traditional Newton iteration-related methods (we use method [8] as a representative). Generally speaking, the method [27] avoids the sensitivity to initial value of traditional Newton iteration-related methods. The process of calculation of Wang et al. [27] method should be divided into a lot of tracing segments. For example, if the output of the first line tracing along the base surface is P_n^1, then P_n^1 will be the initial value of the second line segment. The line segment of PP_n^1 is described as $L_2(t)$ as follows:

$$L_2(t) = tP + (1-t)P_n^1, \quad 0 \le t \le 1 \tag{38}$$

Afterwards, trace the base surface along L_2 with a small iteration step and repeat the tracing loop, until inequality (29) is satisfied. The repeated tracing along a curve on surface is time-killing.

For the trajectory of composite fibre path planning, a large number of points often need to be processed continuously on a given surface. That is a typical multipoint inversion problem. So successful calculation at one stroke is very important. We cannot afford even a few iteration failures, which, as we know, may not be avoided in most Newton-iteration-related methods. For example, in order to improve success rate of Hu's second-order method, when the base surface is NURBS, a subdivision strategy to optimize the initial value for this method will be adopted. The main process of segmentation algorithm is to divide the NURBS surface into a lot of subsurfaces. In each subsurface select the parametric point [0.5, 0.5] as initial value. Proceed as follows:

(1) Divide NURBS surface into $(n-k+1) \times (m-k+1)$ pieces of Bezier surfaces via inserting nodes algorithm.

(2) For each piece of Bezier surface, discrete the Bezier surface if P is in its convex hull. Usually, the surface needs to be divided three times.

Table 1: Two different methods in solving multipoint inversion on hyperbolic paraboloid.

	New Method	Wang's Method
Number of points	2000	2000
CPU Time(s)	29.322629806545390	273.7031542564848
Error	10^{-5}	10^{-5}
Failure	0	0
Maximum iteration	200	200
Parametric range	[−300, 300]	[−300, 300]

Table 2: Experiment of new method for hyperbolic paraboloid.

New Method	CPU Time(s)	Failure
Experiment 1 (2000 points)	31.163845291790800	0
Experiment 2 (2000 points)	30.898390837119763	0
Experiment 3 (2000 points)	30.926868387619347	0

5.1.1. Hyperbolic Paraboloid Surface. Hyperbolic paraboloid surfaces are widely used in construction, manufacturing, aerospace and nuclear industries. The hyperbolic paraboloid surface can be described in Cartesian coordinate equation as follows:

$$z = \frac{x^2}{4} - \frac{y^2}{9} \tag{39}$$

Convert (39) to parametric surface as the following characteristic equation:

$$S(u, v) = \left(u, v, \frac{u^2}{4} - \frac{v^2}{9}\right) \tag{40}$$

The calculation initial point P_0 is $(0, 0, 0)$. The 2000 points need to be computed on base surface. See Figure 4(a). Figure 4(b) directly gives the result of single point calculation step for new method. It takes 106 iterations to complete the calculation of this point.

By comparing with Wang's method, the results of calculating hyperbolic paraboloid are showed in Tables 1–3.

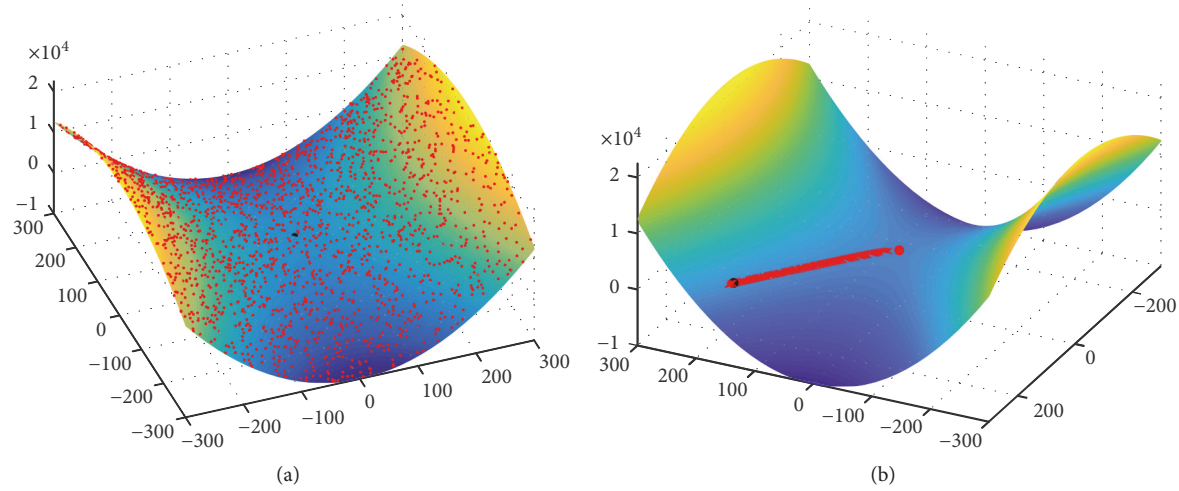

Figure 4: (a) Points on hyperbolic paraboloid. (b) Single point calculation step on hyperbolic paraboloid for new method.

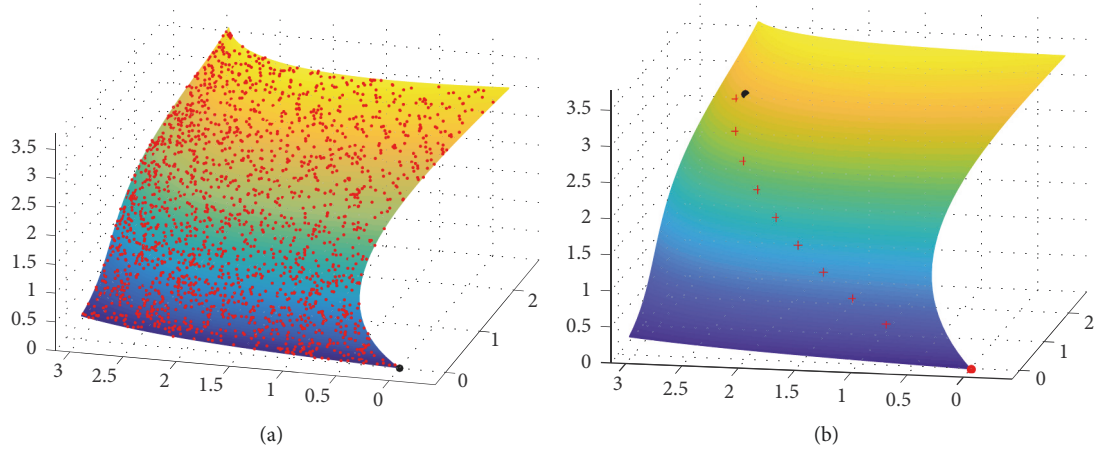

Figure 5: (a) Points on Bezier surface. (b) Single point calculation step on Bezier surface for new method.

Table 3: Experiment of Wang's Method for hyperbolic paraboloid.

Wang's Method	CPU Time(s)	Failure
Experiment 1 (2000 points)	255.5892196051562	0
Experiment 2 (2000 points)	259.5492855696442	0
Experiment 3 (2000 points)	265.7260784417069	0

Table 1 shows the time cost and failure number of points in calculating same 2000 points. From the table we can see that new method costs less time than Wang's method.

Tables 2 and 3 give three times experiment in calculating random 2000 points in parametric domain for three different methods. In each experiment the new method costs less time than other methods and the calculation time has very small fluctuations. Wang's method through multisegment tracing can also have a high success rate.

According to the results, both Wang's method and new method have no failure points. However, the new method is an order of magnitude faster than Wang's method.

5.1.2. Bezier Surface. Another important surface in modern industry is the Bezier surface, it is widely used in surface modelling. A Bezier surface can be expressed as $\boldsymbol{S}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \boldsymbol{b}_{i,j} B_{i,m}(u) B_{j,n}(v)$. The control vertices $\boldsymbol{b}_{i,j}$ is in Table 4.

The initial point $\boldsymbol{P}_0$ is $(0, 0, 0)$ and the 2000 points will be computed on Bezier surface. See Figure 5(a). Figure 5(b) directly gives the result of single point calculation step for new method. It takes 14 iterations to complete the calculation of this point.

As explained in the beginning of this section, Hu et.al [8] elaborates an approach to segment the NURBS surface into Bezier surfaces. A Bezier surface also needs to be segmented into 64 small pieces to optimize initial point for the Newton-Raphson iteration. For this calculation example, the deCasteljau midpoint segmentation algorithm will be applied to solve surface segmentation. According to Hu et al. [31], the detachment times are generally selected as 3. However, the algorithm will cause segmentation error. In order to make sure that every point is in the convex hull

Table 4: Control Points of Bezier Surface.

	$j = 0$	1	2	3
$i = 0$	(0, 0, 0)	(0, 1.116, 0.067)	(0, 2.232, 0.134)	(0.433, 3.132, 0.076)
1	(−0.616, 0.433, 1.217)	(−0.183, 1.333, 1.158)	(−0.183, 2.449, 1.225)	(1.116, 2.915, 0.917)
2	(−0.366, 0.433, 2.183)	(−0.366, 1.549, 2.25)	(−0.366, 2.665, 2.317)	(0.933, 3.132, 2.009)
3	(0.75, 0, 2.900)	(0.75, 1.116, 2.967)	(0.75, 2.232, 3.033)	(2.049, 2.699, 2.725)
4	(1.866, −0.433, 3.616)	(1.433, 0.900, 3.808)	(1.433, 2.016, 3.875)	(2.732, 2.483, 3.567)

Table 5: Three different methods in solving multipoint inversion on Bezier surface.

	New Method	Wang's Method	Hu's second order method
Number of points	2000	2000	2000
CPU Time(s)	6.926983581000674	69.154373822805740	714.2510553485881
Error	10^{-5}	10^{-5}	10^{-5}
Failure	0	0	112
Maximum iteration	200	200	200
Point range	[0, 1]	[0, 1]	[0, 1]

Table 6: Experiment of new method for Bezier surface.

New method	CPU Time(s)	Failure
Experiment 1 (2000 points)	6.962026113531086	0
Experiment 2 (2000 points)	6.546980250216993	0
Experiment 3 (2000 points)	6.702481947743534	0

Table 7: Experiment of Wang's Method for Bezier surface.

Wang's Method	CPU Time(s)	Failure
Experiment 1 (2000 points)	68.42192473628629	0
Experiment 2 (2000 points)	66.11651475984002	0
Experiment 3 (2000 points)	66.90059448796946	0

Table 8: Experiment of Hu's second order method for Bezier surface.

Hu's second order method	CPU Time(s)	Failure
Experiment 1 (2000 points)	691.0428227734579	132
Experiment 2 (2000 points)	667.2941246635530	109
Experiment 3 (2000 points)	665.4855344774451	123

of divided surface, the surface convex hull scope should be expanded.

Table 5 shows that the new method is superior to Wang's method through multisegment tracing and Hu's second-order method with subdivision. Another important phenomenon is that, for Hu's method, the calculation success rate has a great improvement after surface subdivision. But, the subdivision algorithm will cost much more time in every point calculating process.

By randomized numerical experiment in three different methods, Tables 6–8 show that the presented method (new method) costs the least time among the three methods. It also has a great successful rate. Meanwhile, Table 7 shows that Wang's method through multisegment tracing also has

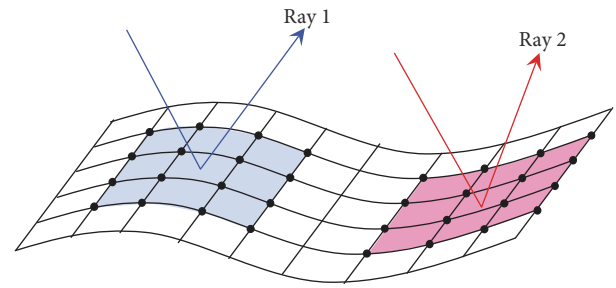

Figure 6: Rays reflecting from freeform surface.

no failure in three numerical experiments. But the CPU time of Wang's method through multisegment tracing is approximately 10 times as much as that of new method. At last, Hu's second order method needs to conduct multilevel subdivision calculation. This is the reason why the time cost by Hu's method is approximately 100 times than that of new method takes. Moreover the failure rate of Hu's method with subdivision is about 6.07%. See Table 8.

Three comparative results of new method, Wang's method through multisegment tracing and Hu's second order method with subdivision, are displayed on Tables 6–8. With the total number of points growing, the CPU time cost using the three algorithms shows different properties. Here we choose a Bezier surface and a hyperbolic paraboloid surface as base surfaces to test the CPU time cost.

5.2. Multiray Surface Intersection. For multiray surface intersection, one of the industrial applications in optical engineering is freeform optical system. However, freeform optics that achieve excellent optical performance have broad application prospects in various areas, such as green energy, manufacturing, illumination, and biomedical [34–38]. The typical feature of such an imaging system is that the object plane or the phase plane is freeform surface. See Figure 6.

Table 9: Control points of Bezier surface screen.

	$j = 0$	1	2	3
$i = 0$	(0, 0, 0)	(0, 20, 10)	(0, 40, 20)	(0, 60, 20)
1	(20, 0, 20)	(20, 20, 20)	(20, 40, 30)	(20, 60, 10)
2	(40, 0, 4)	(40, 20, 30)	(40, 40, 40)	(40, 60, 20)
3	(60, 0, 0)	(60, 20, 10)	(60, 40, 20)	(60, 60, 0)
4	(80, 0, 0)	(80, 20, 0)	(80, 40, 10)	(80, 60, 0)

Table 10: Algorithm parameters and results for the case.

Number of points	CPU Time(s)	Error	Failure	Maximum Iteration	Step of ODE Solver
66848	246.0136530198419	10^{-5}	0	200	0.1

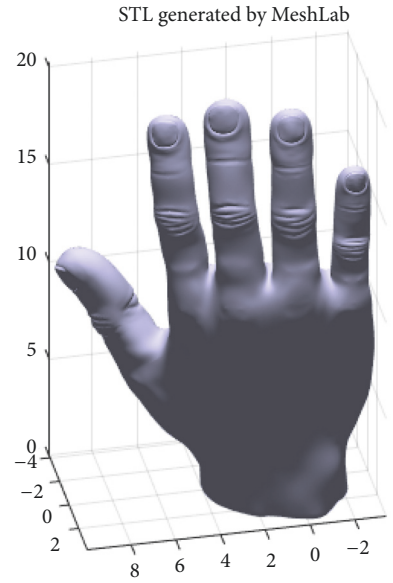

Figure 7: Mesh surface for human hand.

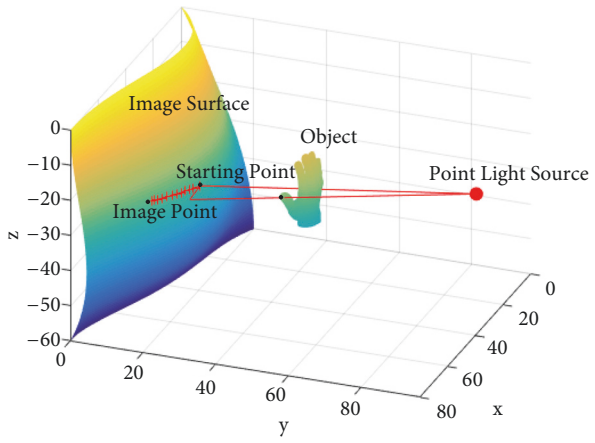

Figure 8: Single point calculation process.

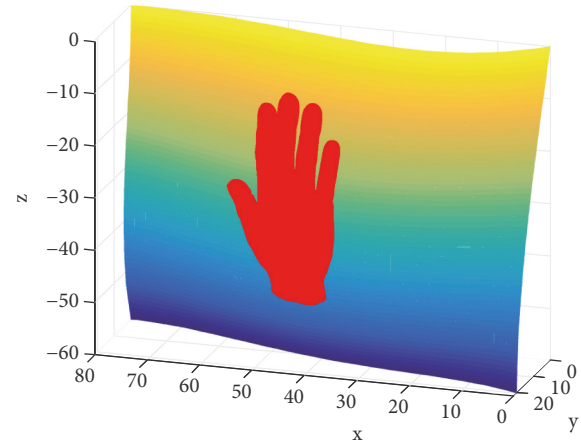

Figure 9: Image of the hand mesh object.

The aberration parameters such as NURBS surface are obtained by controlling the surface shape [39], because it is formed from piecewise splines. According to the geometric properties of the surface and the law of reflection the incident rays and reflected rays can be determined. Therefore, the core algorithm for freeform optical system is the calculation of feature image point which is closely related to multiray surface intersection.

The current case takes a human hand mesh surface as the object which has 66848 points and 133692 faces. See Figure 7. Project the object onto a Bezier surface screen by a point light source. The surface screen is a 3×4-order Bezier surface. The control points are showed in Table 9.

The calculation process for single point and the relative position of the point light source, object and surface screen are showed in Figure 8. The object locates in $(40, 50, -35)$. The light source position is $(39.59, 96.44, -19.78)$. The surface image of the object is shown in Figure 9. The parameters and results of multiray surface intersection algorithm for the case are showed in Table 10.

All the algorithm examples are tested in a MATLAB R2016a on Windows 7 with an Intel 4-core 3.7GHz CPU and 8G RAM.

6. Discussion

The results show that the algorithm combining the Newton-Raphson iteration and the Runge-Kutta method can solve multipoint inversion faster than Wang's and Hu's method. The new method also has a strong stability because the distance between initial point and point to be calculated has an upper bound for a known surface. However, the algorithm through multisegment tracing also shows a great success rate on calculating multipoint inversion. The advantage of this method is no sensitivity to initial value. The method can satisfy tolerance conditions through optimizing the step size of Runge-Kutta algorithm in each time tracing line segment. The second order of Hu's method with subdivision can only be used for calculating point inversion on NURBS surface. The reason for high failure rate is 3 times detachment. If the failure points are recomputed with more detachment times, the failure points may be calculated successfully. But, it will cost more time than Table 8 showed. For multiray surface intersection, the result of present application supports that the algorithm can accurately calculate the imaging under multiray irradiation. In detail, the following advantages and precautions can be presented with the comparison of three algorithms:

(1) The point inversion and ray-surface intersection are very basic problems in computer graphics. There are a lot of algorithms to solve this problem. But, in industry application, engineers often have to face multipoint inversion and multiray surface intersection on base surface. This paper provides a new method to fill this gap.

(2) There is no need to subdivide surface for better initial value. Theoretically speaking, the algorithm absorbs the merit of the classic Newton-Raphson iteration and tracing along the base surface. The cumulative error of Runge-Kutta algorithm determines the initial value of the classic Newton-Raphson iteration. The suitable step size of Runge-Kutta algorithm is the essential key to guarantee the Newton-Raphson iteration convergence.

(3) The algorithm has a large convergence scope. It can solve multipoint inversion and multiray surface intersection not only on the NURBS surface but also on most small curvature open surfaces. In most cases, it can obtain a high success rate.

(4) Similarly to Wang's method, the method needs to be given a curve from initial point to the point to be calculated. If the base surface has a complex topology, we can insert some middle points to be traced on the surface until the initial point is good enough for the Newton-Raphson iteration. For example, if the base surface is a tour patch surface, some intermediate points should be traced when calculating points with no line segment projection on base surface.

(5) The algorithm can also combine tracing along the base surface and Hu's second-order method. The author thinks that the method tracing along the base surface can provide good initial value instead of surface subdivision in contrast to Hu's second-order method.

(6) If the point lies on the boundary of base surface or coincident with the endpoints, the algorithms can also have an excellent calculation success rate.

7. Conclusions

In this paper, we have presented a novel method to solve multipoint inversion and multiray surface intersection problem. This method achieves better results in both accuracy and effectiveness, especially on small curvature open parametric surface. Different from Wang's method, this method combines the tracing along base surface and the classic Newton-Raphson iteration. Better than Hu's method [32], this method need not select initial value in computing every single point inversion. The new approach can save a lot of calculation time and can be used in many industrial applications.

On the other hand, for a NURBS surface, this method significantly increases the success rate of the algorithm compared with Hu's method with three times subdivision. Another important advantage is that the time required to complete calculation is approximately proportional to the number of points on Bezier surface.

For algorithm stabilization, new method has a strong stability on both Bezier surface and hyperbolic parabolic surface. In calculating multipoints inversion problem, Wang's method through multisegmentation tracing also has a good stability, but it costs much more time. Yet, second order of Hu's method with subdivision is the most unstable one on Bezier surface among the three methods.

Finally, the algorithm can also provide excellent support for industrial application. Compared with traditional algorithms, our algorithm can acquire high precision phase in designing freeform optical system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work in this paper was supported by National Natural Science Foundation of China under grant No. 51575266.

References

[1] G. Farin, J. Hoschek, and M.-S. Kim, *Handbook of Computer Aided Geometric Design*, Elsevier, 2002.

[2] T. W. Sederberg, D. C. Anderson, and R. N. Goldman, "Implicit representation of parametric curves and surfaces," *Computer Vision, Graphics, and Image Processing*, vol. 28, no. 1, pp. 72–84, 1984.

[3] C. M. Hoffmann, "Implicit curves and surfaces in CAGD," *IEEE Computer Graphics and Applications*, vol. 13, no. 1, pp. 79–88, 1993.

[4] T. W. Sederberg and R. N. Goldman, "Algebraic geometry for computer-aided geometric design," *IEEE Computer Graphics and Applications*, vol. 6, no. 6, pp. 52–59, 1986.

[5] J. Xu, W. Liu, J. Wu, H. Bian, and L. Li, "Geometric algorithm for point projection and inversion onto bézier surfaces," *Frontiers of Computer Science in China*, vol. 3, no. 4, pp. 472–476, 2009.

[6] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A K Peters, Wellesley, Mass, USA, 1993.

[7] E. Hartmann, "On the curvature of curves and surfaces defined by normalforms," *Computer Aided Geometric Design*, vol. 16, no. 5, pp. 355–376, 1999.

[8] S.-M. Hu and J. Wallner, "A second order algorithm for orthogonal projection onto curves and surfaces," *Computer Aided Geometric Design*, vol. 22, no. 3, pp. 251–260, 2005.

[9] X.-M. Liu, L. Yang, J.-H. Yong, H.-J. Gu, and J.-G. Sun, "A torus patch approximation approach for point projection on surfaces," *Computer Aided Geometric Design*, vol. 26, no. 5, pp. 593–598, 2009.

[10] L. A. Piegl and W. Tiller, "Parametrization for surface fitting in reverse engineering," *Computer-Aided Design*, vol. 33, no. 8, pp. 593–603, 2001.

[11] Y. L. Ma and W. T. Hewitt, "Point inversion and projection for NURBS curve and surface: control polygon approach," *Computer Aided Geometric Design*, vol. 20, no. 2, pp. 79–99, 2003.

[12] D. E. Johnson and E. Cohen, "Framework for efficient minimum distance computations," in *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3678–3684, IEEE, 1998.

[13] F. Chin and C. A. Wang, "Optimal algorithms for the intersection and the minimum distance problems between planar polygons," *IEEE Transactions on Computers*, vol. C-32, no. 12, pp. 1203–1207, 1983.

[14] H. Edelsbrunner, "Computing the extreme distances between two convex polygons," *Journal of Algorithms*, vol. 6, no. 2, pp. 213–224, 1985.

[15] T. Park, J. Ji, and K. H. Ko, "A second order geometric method for ray/parametric surface intersection," *Computer Aided Geometric Design*, vol. 30, no. 8, pp. 795–804, 2013.

[16] S.-W. Wang, Z.-C. Shih, and R.-C. Chang, "An efficient and stable ray tracing algorithm for parametric surfaces," *Journal of Information Science and Engineering*, vol. 18, no. 4, pp. 541–561, 2002.

[17] S.-W. Wang, Z.-C. Shih, and R.-C. Chang, "An improved rendering technique for ray tracing Bézier and B-spline surfaces," *Journal of Visualization and Computer Animation*, vol. 11, no. 4, pp. 209–219, 2000.

[18] D. Saini and S. Kumar, "Stereo vision-based conic reconstruction using a ray-quadric intersection," *International Journal of Image and Graphics*, vol. 15, no. 4, Article ID 1550019, 2015.

[19] M. E. Mortenson, *Geometric Modeling*, 1997.

[20] C. H. Song, X. Xu, K. L. Shi et al., "Projecting points onto planar parametric curves by local biarc approximation," *Computers & Graphics*, vol. 38, pp. 183–190, 2014.

[21] K. Ko and T. Sakkalis, "Orthogonal projection of points in CAD/CAM applications: an overview," *Journal of Computational Design and Engineering*, vol. 1, no. 2, pp. 116–127, 2014.

[22] H.-Y. Xu, X. Fang, H.-y. Tam, X. Wu, and L. Hu, "A second-order algorithm for curve orthogonal projection onto parametric surface," *International Journal of Computer Mathematics*, vol. 89, no. 1, pp. 98–111, 2012.

[23] A. Limaiem and F. Trochu, "Geometric algorithms for the intersection of curves and surfaces," *Computers and Graphics*, vol. 19, no. 3, pp. 391–403, 1995.

[24] M. C. Lin and D. Manocha, "Fast interference detection between geometric models," *The Visual Computer*, vol. 11, no. 10, pp. 542–561, 1995.

[25] I. Hanniel, A. Krishnamurthy, and S. McMains, "Computing the hausdorff distance between NURBS surfaces using numerical iteration on the GPU," *Graphical Models*, vol. 74, no. 4, pp. 255–264, 2012.

[26] Y.-T. Oh, Y.-J. Kim, J. Lee, M.-S. Kim, and G. Elber, "Continuous point projection to planar freeform curves using spiral curves," *The Visual Computer*, vol. 28, no. 1, pp. 111–123, 2012.

[27] X. Wang, W. Zhang, and X. Huang, "Computation of point inversion and ray-surface intersection through tracing along the base surface," *The Visual Computer*, vol. 31, no. 11, pp. 1487–1500, 2015.

[28] Y. Oh, Y. Kim, J. Lee, M. Kim, and G. Elber, "Efficient point-projection to freeform curves and surfaces," *Computer Aided Geometric Design*, vol. 29, no. 5, pp. 242–254, 2012.

[29] X. Chen and W. Ma, "Rational cubic clipping with linear complexity for computing roots of polynomials," *Applied Mathematics and Computation*, vol. 273, pp. 1051–1058, 2016.

[30] M. Lotkin, "On the accuracy of Runge-Kutta's method," *Mathematics of Computation*, vol. 5, pp. 128–133, 1951.

[31] S. Hu, J. Sun, T. Jin, and G. Wang, "Computing the parameters of points on Nurbs curves and surfaces via moving affine frame method," *Journal of Software*, vol. 11, no. 1, pp. 49–53, 2000.

[32] L. Piegl and W. Tiller, *The NURBS book*, Springer Science & Business Media, 2012.

[33] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, USA, 1970.

[34] F. Z. Fang, X. D. Zhang, A. Weckenmann, G. X. Zhang, and C. Evans, "Manufacturing and measurement of freeform optics," *CIRP Annals - Manufacturing Technology*, vol. 62, no. 2, pp. 823–846, 2013.

[35] R. Wu, H. Wang, P. Liu et al., "Efficient optimal design of smooth optical freeform surfaces using ray targeting," *Optics Communications*, vol. 300, pp. 100–107, 2013.

[36] A. Bäuerle, A. Bruneton, R. Wester, J. Stollenwerk, and P. Loosen, "Algorithm for irradiance tailoring using multiple freeform optical surfaces," *Optics Express*, vol. 20, no. 13, pp. 14477–14485, 2012.

[37] K. Fuerschbach, J. P. Rolland, and K. P. Thompson, "Theory of aberration fields for general optical systems with freeform surfaces," *Optics Express*, vol. 22, no. 22, pp. 26585–26606, 2014.

[38] L. Zhang, D. Huang, W. Zhou, C. Fan, S. Ji, and J. Zhao, "Corrective polishing of freeform optical surfaces in an off-axis three-mirror imaging system," *The International Journal of Advanced Manufacturing Technology*, vol. 88, no. 9-12, pp. 2861–2869, 2017.

[39] M. P. Chrisp, B. Primeau, and M. A. Echter, "Imaging freeform optical systems designed with NURBS surfaces," *Optical Engineering*, vol. 55, no. 7, Article ID 071208, 2016.