Singularity Computation for Rational Parametric Surfaces Using Moving Planes

XIAOHONG JIA, KLMM, AMSS, Chinese Academy of Sciences and University of Chinese Academy of Sciences, China
FALAI CHEN, University of Science and Technology of China
SHANSHAN YAO, Beijing University of Posts and Telecommunications, China and University of Science and Technology of China

Singularity computation is a fundamental problem in Computer Graphics and Computer Aided Geometric Design, since it is closely related to topology determination, intersection, mesh generation, rendering, simulation, and modeling of curves and surfaces. In this article, we present an efficient and robust algorithm for computing all the singularities (including their orders) of rational parametric surfaces using the technique of moving planes. The main approach is first to construct a representation matrix whose columns correspond to moving planes following the parametric surface. Then, by substituting the parametric equation of the rational surface into this representation matrix, one can extract the singularity information from the corresponding matrix and return all the singular loci including self-intersection curves, cusp curves, and isolated singular points of the rational surface, together with the order of each singular locus. We present some examples to compare our algorithm with state-of-the-art methods from different perspectives including robustness, efficiency, order computation, and numerical stability, and the experimental results show that our method outperforms existing methods in all these aspects. Furthermore, applications of our algorithm in surface rendering, mesh generation and surface/surface intersections are provided to demonstrate that correctly computing the self-intersection curves of a surface is essential to generate high quality results for these applications.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Shape modeling**; **Parametric curve and surface models**; **Symbolic and algebraic manipulation**; **Symbolic and algebraic algorithms**; **Algebraic algorithms;**

Additional Key Words and Phrases: Rational parametric surface, singularity, implicitization matrix, moving plane

ACM Reference format:
Xiaohong Jia, Falai Chen, and Shanshan Yao. 2022. Singularity Computation for Rational Parametric Surfaces Using Moving Planes. *ACM Trans. Graph.* 42, 1, Article 12 (September 2022), 14 pages.
https://doi.org/10.1145/3551387

The work is supported by National Key R&D Program of China (2021YFB1715900), National Natural Science of Foundation of China for Outstanding Young Scholars (12022117) and National Natural Science of Foundation of China (61972368, 61872354).
Authors' addresses: X. Jia, KLMM, AMSS, Chinese Academy of Sciences, Beijing and University of Chinese Academy of Sciences, Beijing, China; email: xhjia@amss.ac.cn; F. Chen (corresponding author), University of Science and Technology of China, Hefei, Anhui; email: chenfl@ustc.edu.cn; S. Yao, Beijing University of Posts and Telecommunications, Beijing, China and University of Science and Technology of China, Hefei, Anhui; email: ssyao@ustc.edu.cn.

0730-0301/2022/09-ART12 $15.00
https://doi.org/10.1145/3551387

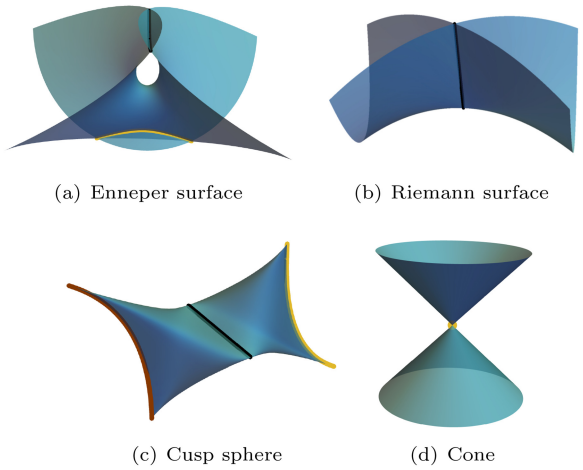


Fig. 1. Singular points on rational surfaces: (a) Enneper surface with two self-intersection curves; (b) Riemann surface with a self-intersection curve; (c) Cusp sphere with two cusp curves at both sides and a self-intersection curve in the middle; (d) A cone with an isolated singular point.

1 INTRODUCTION

Singular points, or *singularities*, are those points where the tangent lines or planes are not uniquely defined on curves or surfaces. Figure 1 illustrates self-intersection curves, cusp curves, and isolated singular points of several rational parametric surfaces. Since singular points are special features of curves and surfaces, their detection is fundamental to many crucial applications in Computer Graphics and geometric modeling, including mesh generation [de Araújo et al. 2015; Li and Barbič 2018; Plantinga and Vegter 2004; Zhou et al. 2016], surface rendering [Barringer et al. 2012; Harbinson et al. 2019; Loop and Blinn 2006], surface intersection [Krishnan and Manocha 1997; Lin et al. 2014; Pekerman et al. 2008], topology determination [Gueziec et al. 2001; Wang 1981], simulation [Baraff et al. 2003; Wong et al. 2018] and CNC machining [Hoschek et al. 1993; Xu et al. 2015].

Detecting singularities of rational curves or surfaces includes computing the pre-images of singular points in the parametric space and determining the orders of singular points. There has been extensive research on computing and classifying singular points of rational curves [Chen et al. 2008; Jia and Goldman 2009, 2012; Li and Cripps 1997; Manocha and Canny 1992; Pérez-Díaz 2007; Peterson 1917; Sakai 1999; Sakkalis and Farouki 1990; Shi and Chen 2010; Shi et al. 2013; Wang et al. 2009]. Nevertheless, fast and robust computation of singularities of rational surfaces

has remained a challenging problem in the geometric modeling community [Busé et al. 2016; Chen et al. 2005].

Existing work on computing the singularities of rational surfaces is mainly based on symbolic approaches, including algebraic decomposition [Elber et al. 2009; Pekerman et al. 2008], resultant construction [Busé et al. 2008], Gröbner bases [Huang and Wang 2011], μ-bases [Jia et al. 2009] and characteristic sets [Pérez-Díaz et al. 2015]. These methods have different restrictions, such as losing cusp curves or isolated singular points, giving extraneous factors, or suffering from inefficiency. More importantly, symbolic methods tend to fail in numerical computations.

In this article, we introduce an efficient and robust symbolic approach for computing the singularities of rational surfaces. Our method computes not only self-intersection points but also cusps and isolated singular points, and not only the pre-images of singular points in the parametric space, but also the exact orders of the singular points. The result neither misses any singular point nor produces any extraneous factor. Furthermore, our algorithm is impervious to base points and it works well for any type of rational parametrization. Finally, our approach is numerically stable in floating point computations.

The main ingredient in our approach is the use of moving planes that follow a given surface parametrization. The technique of moving lines and moving planes has been successfully applied in implicitization [Chen et al. 2005; Sederberg and Chen 1995; Sederberg et al. 1997; Shen and Goldman 2017] and singularity computation of rational curves and some special types of surfaces [Chen et al. 2008; Jia et al. 2009; Jia and Goldman 2009, 2012; Shi and Chen 2010; Shi et al. 2013; Wang et al. 2009]. Our method is based on an important result from Busé [2014], where implicit matrix representations of rational parametric surfaces are constructed by computing a set of moving planes following a given parametric surface. From the rank deficiency of the implicit matrix representations, we present an elegant algorithm to extract the singularities of general rational parametric surfaces.

In summary, the main contributions of the current work include:

- An efficient and robust algorithm for computing the singularities of rational surfaces. Compared with existing symbolic approaches, our algorithm is numerically stable and is impervious to base points, that is, it works for any type of rational parameterizations.
- The proposed algorithm computes not only self-intersection curves, but also cusp curves and isolated singular points that are often missed by existing methods. The orders of the singularities are also computed.
- Applications of the algorithm in surface rendering, mesh generation, and surface/surface intersections are provided to demonstrate the importance of singularity computations.

2 RELATED WORK

We next briefly review related work on singularity computation for rational curves and surfaces.

2.1 Computing Singular Points of Rational Curves

There is a rich literature for computing singular points of rational curves. Sakkalis and Farouki [1990] describe a method to determine whether an algebraic curve is singular, and if so, how to isolate the singular points including multiplicities, and how to count the number of distinct tangents at each singular point. Sakai [1999] presents an algorithm for computing the inflection points and singularities of planar rational cubic curve segments. Pérez-Díaz [2007] provides a method for computing the singularities of rational planar curves and analyzing the properties of non-ordinary curves.

The technique of moving lines and moving planes has also been employed for fast computation of singularities for rational curves. Chen et al. [2008] show that μ-bases are deeply tied to the singular points of rational planar curves and provide an efficient symbolic algorithm to compute the entire singularity trees of rational planar curves. Jia and Goldman [2009; 2012] and Busé and D'Andrea [2012] further explore the algebraic relationship between singularities and μ-bases for rational planar curves. Shi et al. [2013] provide symbolic algorithms for computing singularities of rational space curves. Some other work focuses only on detecting cusps and inflection points in curves [Gueziec et al. 2001; Manocha and Canny 1992].

2.2 Singularity Computation for Surfaces

Existing work on computing singular points of rational surfaces is relatively rare. Elimination methods, such as resultants and Gröbner bases, have been used in finding the singularities of rational surfaces.

For example, Busé et al. [2008] extract the parameters of singular points through the Bézout matrix constructed from a parametric surface. This approach is generally efficient, but is invalid when there are base points on the parametric surface. Furthermore, the method cannot compute cusps and the results sometimes contain extraneous factors or unnecessary multiples of singular factors.

Huang and Wang [2011] present two approaches to compute the self-intersection curves of rational surfaces, one based on regular systems and the other based on Gröbner bases. Due to the symbolic computation involving Gröbner bases or regular systems, their approach becomes inefficient as the degree of the rational surface increases. Furthermore, it can deal only with self-intersection curves but not cusps, and a proper parametrization of the rational surface is required.

Pérez-Díaz et al. [2015] computes all the singular points of rational parametric surfaces in projective space, together with their orders. Moreover, in their setting, the parametrization does not need to be proper and the surface can contain base points. However, due to the computation of regular systems, their approach also suffers from the problem of computational inefficiency.

The technique of moving planes is a useful tool to analyze and compute the singularities of rational surfaces. For example, Jia et al. [2009] provide an efficient algorithm to calculate the singular curves together with their orders of rational ruled surfaces by μ-bases. Wang and Chen [2012] compute the singularities of a Steiner surface by employing moving planes following the Steiner surface. Botbol et al. [2014] further explore the relationship of singular points and syzygies of rational surfaces by using commutative algebra. Despite all this work, there still lacks an

efficient, robust, numerically stable and universal algorithm for computing singularities for general rational surfaces.

3 MATRIX REPRESENTATION OF RATIONAL SURFACES

We first provide some preliminaries on rational surfaces and moving planes, and then introduce the concept of matrix representations.

3.1 Rational Surfaces

A **rational parametric surface** in $\mathbb{R}^3$ can be represented by a map

$$\begin{aligned} \phi:\ & \mathbb{R}^2 \dashrightarrow \mathbb{R}^3 \\ & (s,t) \longrightarrow \left(\frac{f_0(s,t)}{f_3(s,t)}, \frac{f_1(s,t)}{f_3(s,t)}, \frac{f_2(s,t)}{f_3(s,t)}\right), \end{aligned} \tag{1}$$

where $f_i, i = 0,\ldots,3$ are polynomials that are linearly independent and satisfy $\gcd(f_0, f_1, f_2, f_3) = 1$. The parametric Equation (1) can be a tensor-product surface of bi-degree (d_1, d_2) in s, t, i.e., the polynomials $f_i, i = 0,\ldots,3$ have maximum degree d_1 in s and d_2 in t; or a rational triangular surface of degree d, i.e., the polynomials $f_i, i = 0,\ldots,3$ have maximum total degree d in (s,t). To allow for points at infinity, throughout this article we shall adopt the homogeneous parametrization Φ of ϕ in 3D projective space:

$$\begin{aligned} \Phi:\ & \mathbb{R}^2 \dashrightarrow \mathbb{PR}^3 \\ & (s,t) \longrightarrow (f_0(s,t), f_1(s,t), f_2(s,t), f_3(s,t)). \end{aligned} \tag{2}$$

For example, a rational parametrization ϕ of a sphere is

$$\phi(s,t) = \left(\frac{2(1-s^2)t}{(1+s^2)(1+t^2)}, \frac{(1-s^2)(1-t^2)}{(1+s^2)(1+t^2)}, \frac{2s}{(1+s^2)}\right). \tag{3}$$

The homogeneous parametrization Φ of this sphere is

$$\Phi(s,t) = (2(1-s^2)t, (1-s^2)(1-t^2), 2s(1+t^2), (1+s^2)(1+t^2)). \tag{4}$$

Under the homogeneous parametrization, a point $(x_0, y_0, z_0) \in \mathbb{R}^3$ is written as $(x_0, y_0, z_0, 1) \in \mathbb{PR}^3$, while a point at infinity has the form $(x_0, y_0, z_0, 0) \in \mathbb{PR}^3$. To account for parameter values at infinity, the parameters s, t in Equation (2) should also be homogenized as $s : u$ and $t : v$ in real projective space. However, for brevity of description, we still use the affine form in Equation (2) in the following discussion. In this case, we allow $s = \infty$ (corresponding to $1 : 0$) or $t = \infty$.

A (complex) parameter pair (s_0, t_0) is called a *base point* of the rational parametrization Φ if $\Phi(s_0, t_0) = (0, 0, 0, 0)$. For example, $(1, i)$, $(-1, i)$,$(1, -i)$, $(-1, -i)$ are four base points of the parametrization of the sphere (4).

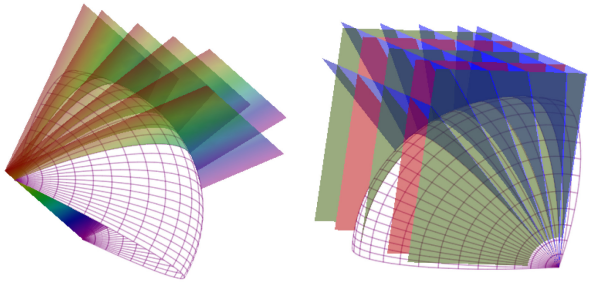
3.2 Moving Planes

A *moving plane* is a family of planes with parameters s, t:

$$L(s,t;X) := l_0(s,t)x + l_1(s,t)y + l_2(s,t)z + l_3(s,t)w = 0,$$

where $l_i(s,t), i = 0,\ldots,3$ are bivariate polynomials, called *blending functions*. Sometimes for brevity, we also denote the moving plane L by its vector form

$$\mathbf{L}(s,t) := (l_0(s,t), l_1(s,t), l_2(s,t), l_3(s,t)).$$



(a) (b)

Fig. 2. (a) The moving plane $L_1(t;X) := (1-t^2)x - 2ty = 0$ for $t = 0, 0.1, 0.3, 0.5, 0.7, 0.95$ that follows a parametric sphere. (b) The intersection of two moving planes $L_1(t;X) = 0$ for $t = 0, 0.1, 0.3, 0.5, 0.7, 0.95$ and $L_2(s;X) = (s^2+1)z - 2sw = 0$ for $s = -0.8, -0.3, 0, 0.3, 0.8$.

A moving plane $L(s,t;X) = 0$, or $\mathbf{L}(s,t)$, is said to *follow* the rational parametrization (2) if

$$\mathbf{L}(s,t) \cdot \Phi(s,t) = \sum_{i=0}^{3} l_i(s,t) f_i(s,t) \equiv 0. \tag{5}$$

Geometrically, this equation means that for every pair of parameter values (s,t), the plane $L(s,t;X) = 0$ always passes through the corresponding point $\Phi(s,t)$ on the surface.

For example, $L_1(t;X) := (1-t^2)x - 2ty = 0$ and $L_2(s;X) := (s^2+1)z - 2sw = 0$ are two moving planes that follow the parameterization (4), as shown in Figure 2. The intersection of $L_1(t;X) = 0$ and $L_2(s;X) = 0$ is a moving line $l(s,t)$ that passes through the point $\Phi(s,t)$ on the sphere at every (s,t), as shown in Figure 2(b).

3.3 Matrix Representations

A matrix representation, which provides information on the implicit equation of a parametric surface, can be constructed from a series of moving planes. For example, there are 12 linearly independent moving planes of bi-degree $(1,3)$ in (s,t) that follow the parametric sphere (4):

$$\begin{aligned} L_1 &:= (-s + st^2)x + 2sty, \\ L_2 &:= (t^2-1)x + 2ty, \\ L_3 &:= st^2x - t^3z + st^3(-y+w), \\ L_4 &:= -t^2x + t^3(y+w) - zst^3, \\ L_5 &:= (st + st^3)x - 2t^2z + 2st^2w, \\ &\vdots \end{aligned}$$

Arrange the coefficients of these moving planes in the monomial basis $[1, s, t, st, t^2, st^2, t^3, st^3]$ in a matrix:

$$\mathbb{M}(X) := \begin{pmatrix} 0 & -x & 0 & 0 & 0 & \cdots \\ -x & 0 & 0 & 0 & 0 & \cdots \\ 0 & 2y & 0 & 0 & 0 & \cdots \\ 2y & 0 & 0 & 0 & x & \cdots \\ 0 & x & 0 & -x & -2z & \cdots \\ x & 0 & x & 0 & 2w & \cdots \\ 0 & 0 & -z & y+w & 0 & \cdots \\ 0 & 0 & -y+w & -z & x & \cdots \end{pmatrix}_{8\times 12}. \tag{6}$$

Each column of this matrix comes from a moving plane. The GCD of all the 8×8 minors of the matrix $\mathbb{M}(X)$ is $(x^2+y^2+z^2-w^2)^2$, which is the implicit equation of the parametric sphere raised to the power of 2. The matrix $\mathbb{M}(X)$ is called a *matrix representation* of the parametric sphere.

Such a matrix representation exists for general rational surfaces [Busé 2014]. For rational surfaces of bi-degree (d_1, d_2), the matrix representation can be constructed from all the linearly independent moving planes

$$L(s,t;X) = l_0(s,t)x + l_1(s,t)y + l_2(s,t)z + l_3(s,t)w = 0$$

with bi-degree $(\nu_1, \nu_2) := (2d_1 - 1, d_2 - 1)$ or $(d_1 - 1, 2d_2 - 1)$ in (s,t) that satisfy (5). This results a linear system where all the coefficients in (s,t) of $l_i(s,t)$, $i = 0,1,2,3$ are unknowns. Solving this linear system gives all the bi-degree (ν_1, ν_2) moving planes.

Rewrite all these linearly independent moving planes in the monomial basis in (s,t) as

$$[L_1, \ldots, L_k] = [1, s, \ldots, s^{\nu_1}, t, st, \ldots, s^{\nu_1}t^{\nu_2}] \cdot \mathbb{M}(X),$$

where

$$\mathbb{M}(X) := \begin{pmatrix} \Lambda_{1,1} & \Lambda_{1,2} & \cdots & \Lambda_{1,k} \\ \Lambda_{2,1} & \Lambda_{2,2} & \cdots & \Lambda_{2,k} \\ \vdots & \vdots & & \vdots \\ \Lambda_{q,1} & \Lambda_{q,2} & \cdots & \Lambda_{q,k} \end{pmatrix}_{q\times k}, \tag{7}$$

where $q = (\nu_1 + 1)(\nu_2 + 1) = 2d_1d_2$. Then $\mathbb{M}(X)$ is a matrix representation of the rational surface $\Phi(s,t)$.

Note that if the rational surface is of triangular form of total degree d, then the matrix representation can be constructed similarly from moving planes of total degree $\nu := 2(d-1)$. In the following discussion, we describe our results and algorithms only for tensor-product surfaces since all the statements hold as well for triangular surfaces. See Busé [2014] and Botbol et al. [2014] for details.

Lemma 3.1 ([Busé 2014]). *In the matrix $\mathbb{M}(X)$, it is always true that $q \le k$. Moreover, if a point $\mathbf{Q} = (x_0, y_0, z_0, w_0) \in \mathbb{PR}^3$ is a point on the rational surface $\Phi(s,t)$, then* $\operatorname{rank}(\mathbb{M}(\mathbf{Q})) \le q - 1$.

For example, the point $\mathbf{Q} = (0,0,1,1)$ lies on the rational sphere and $\operatorname{rank}(\mathbb{M}(\mathbf{Q})) < 8$ for the matrix representation (6).

4 SINGULARITIES

Singular points on parametric surfaces are those multiple traced points that have more than one corresponding parameter pairs counted with multiplicity. For a point $\mathbf{Q} = (x_0, y_0, z_0, w_0) \in \mathbb{PR}^3$, let $N_\mathbf{Q}$ be the number of parameter pairs (s,t) (counting with multiplicity) such that $\Phi(s,t) = \kappa \cdot \mathbf{Q}$ for some non-zero constant κ. If $N_\mathbf{Q} = 1$, then the point $\mathbf{Q}$ is called a *simple point*; otherwise, if $N_\mathbf{Q} > 1$, then $\mathbf{Q}$ is a singular point. Furthermore, if $N_\mathbf{Q}$ is finite, we define $N_\mathbf{Q}$ as the order of the point $\mathbf{Q}$; otherwise, if $N_\mathbf{Q} = \infty$, we define the order of $\mathbf{Q}$ as ∞. Figure 3 illustrates three different types of isolated singular points whose orders are ∞.

For a rational parametric surface, the singularities consist of a finite number of one-dimensional branches (such as self-intersection curves and/or cusp curves) and a finite number of isolated singular points. For each isolated singular point, there are an infinite number of corresponding parameter pairs, while for each generic singular point on a one-dimensional singular branch, there are a finite number (which is exactly the order) of corresponding parameter pairs. The number of non-generic singular points is finite. For a more detailed discussion about singularities of surfaces, the reader is referred to Hartshorne [1977].

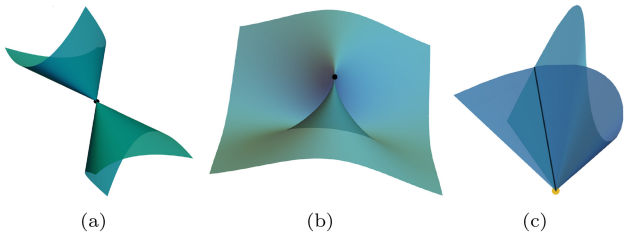


Fig. 3. Isolated singular points whose orders are ∞. (a) The vertex of a cone; (b) An isolated singular point on a parametric surface; (c) An isolated singular point on a self-intersection curve.

4.1 Singular Points in Matrix Representation

Lemma 4.1 ([Busé 2014]). *Suppose that the given rational surface (2) has a matrix representation $\mathbb{M}(X)$ of size $q\times k$, $q \le k$. Let $\mathbf{Q} \in \mathbb{PR}^3$ be a point on the surface. Then $\mathbf{Q}$ is a singular point if and only if* $\operatorname{rank}(\mathbb{M}(\mathbf{Q})) < q - 1$.

Example 4.2. Consider a bi-degree $(2,2)$ rational surface with a parametrization Φ given by

$$\begin{aligned} f_0 &= (2t^2 - 6t + 4)s^2 + 4(t-1)s + 1, \\ f_1 &= -2(t^2 - 3t + 1)s^2 + (4t - 2)s, \\ f_2 &= (t^2 - 4t + 3)s^2 - 2(t-1)^2 s, \\ f_3 &= 1. \end{aligned} \tag{8}$$

Consider all the bi-degree $(\nu_1, \nu_2) = (2d_1 - 1, d_2 - 1) = (3,1)$ moving planes following the parameterization Φ. By solving the system of linear Equations (5), we get a matrix representation $\mathbb{M}(X)$ of Φ:

$$\begin{pmatrix} 0 & 0 & -2x - 2y + 2w & \cdots \\ 0 & 0 & -x - y - \frac{1}{3}w & \cdots \\ 0 & 0 & \frac{2}{3}w & \cdots \\ 0 & 0 & x + y - w & \cdots \\ -2x - 2y + 2w & 3x + 3y - 3w & 0 & \cdots \\ -x - y - \frac{1}{3}w & 2y + 2x & 0 & \cdots \\ \frac{2}{3}w & x + y - w & 0 & \cdots \\ x + y - w & 0 & 0 & \cdots \end{pmatrix}_{8\times 10}.$$

By Lemma 4.1, the point $\mathbf{Q}_0 = (5, -4, -3, 1)$ is a simple point on the surface since $\operatorname{rank}(\mathbb{M}(\mathbf{Q}_0)) = 7$. The points $\mathbf{Q}_1 = (1, 4, 0, 1)$ and $\mathbf{Q}_2 = (1, 28 - 20\sqrt{2}, 0, 1)$ are singular points since $\operatorname{rank}(\mathbb{M}(\mathbf{Q}_1)) = 6$ and $\operatorname{rank}(\mathbb{M}(\mathbf{Q}_2)) = 5$. □

The matrix representation not only tells whether or not a point is singular, but also tells the order of the point, if the point is not an isolated singular point. The following result can be derived from Busé [2014].

Proposition 4.3. *If a point $\mathbf{Q} \in \mathbb{PR}^3$ is not an isolated singular point on the surface, then the order of $\mathbf{Q}$ is* $q - \operatorname{rank}(\mathbb{M}(\mathbf{Q}))$.

Example 4.2 (Continued). The point $\mathbf{Q}_1$ has order $8 - 6 = 2$, and the point $\mathbf{Q}_2$ has order $8 - 5 = 3$.

4.2 Computing Singular Factors

Next, we show how to compute all the singular points on a rational surface from the matrix representation $\mathbb{M}(X)$. Denote by

$$\mathbb{N}(s,t) = \mathbb{M}(f_0(s,t), f_1(s,t), f_2(s,t), f_3(s,t)). \tag{9}$$

Lemma 4.1 implies the following results.

Proposition 4.4. *A point $\Phi(s_0,t_0)$ is a singular point on the parametric surface $\Phi(s,t)$ if and only if (s_0,t_0) is a common root of all the $(q-1)\times(q-1)$ minors of the matrix $\mathbb{N}(s,t)$.*

There are $p = \binom{q}{q-1}\binom{k}{q-1}$ minors of order $q-1$ in $\mathbb{N}(s,t)$. We denote these minors by $N_i(s,t), i = 1,\ldots,p$, and let $H(s,t)$ be their greatest common divisor. From Proposition 4.4, we immediately have:

Proposition 4.5. *If (s_0,t_0) is a root of $H(s,t) = 0$, then $\Phi(s_0,t_0)$ is a singular point of the parametric surface $\Phi(s,t)$.*

Remark 1. There may exist a finite number of parameter pairs $(s_i,t_i)_{i=1}^{l}$ that are common (complex) roots of $N_i(s,t), i = 1,\ldots,p$, but are not roots of $H(s,t)$. However, these parameter pairs do not correspond to any new singular points in 3D space other than the singularities defined by $H(s,t) = 0$, that is, the set $S = \{\phi(s,t) \mid H(s,t) = 0\}$ contains all the singularities of the parametric surface. (See the supplemental file[1] for a detailed proof.) Thus. ignoring $(s_i,t_i)_{i=1}^{l}$ does not influence the computation of singularities.

Therefore, we can factor $H(s,t)$ into the product of irreducible factors $H(s,t) = \prod_{i=1}^{N} h_i(s,t)^{l_i}$, and from each factor $h_i(s,t)$ we can compute the corresponding singularities. We call $h_i(s,t), i = 1,\ldots,N$ the *singular factors* of the rational surface $\Phi(s,t)$.

Example 4.2 (Continued). Substitute $X = (f_0,f_1,f_2,f_3)$ into the matrix $\mathbb{M}(X)$ to obtain the matrix $\mathbb{N}(s,t)$. The GCD of all the 7×7 minors of $\mathbb{N}(s,t)$ is $H(s;t) = h_1^2h_2h_3h_4$, where:

$$\begin{aligned}
h_1 &= s,\\
h_2 &= t-1,\\
h_3 &= s^2+4s-4,\\
h_4 &= 4s^3t^2-5s^3t-3s^3+8s^2t^3-28s^2t^2+28s^2t\\
&\quad +20s^2-4t^3s+20st^2-21st-23s+6-2t.
\end{aligned}$$

Then $h_i = 0, i = 1,\ldots,4$ are all the singular factors of the rational surface. Figure 4 presents a local illustration of the self-intersection curves and the isolated singular points corresponding to these singular factors.

Example 4.6. The singular factors for the rational surfaces shown in Figure 1 are as follows:

- (Enneper). The rational parametric equation is:
$$\Phi(s,t) = \left(s-\frac{s^3}{3}+st^2, t-\frac{t^3}{3}+s^2t, s^2-t^2, 1\right).$$
The singular factors corresponding to the self-intersection curves are: $h_1 = 3s^2-t^2+3$ (black curve) and $h_2 = s^2-3t^2-3$ (yellow curve).
- (Riemann). The rational parametric equation is:
$$\Phi(s,t) = (st, t^2-s^2, 30s, 1).$$
The singular factor corresponding to the self-intersection curve is: $h = s$.
- (Cusp sphere). The rational parametric equation is:
$$\Phi(s,t) = ((2(1-s^2))(1-t^2)s, 2(1-s^2)^2t, 8s^2t, 1).$$
The singular factors corresponding to the two cusp curves are: $h_1 = t+1$ (yellow curve) and $h_2 = t-1$ (red curve). The singular factor corresponding to the self-intersection curve in the middle is $h_3 = t$ (black curve).
- (Cone). The rational parametric equation is:
$$\Phi(s,t) = ((1-s^2)t, 2st, (2t^2-1)(1+s^2), 1+s^2).$$
The singular factor corresponding to the isolated singular point is $h = s$.

4.3 Computing Orders of Singular Factors

Definition 4.7. The *order of a singular factor* $h(s,t)$ is defined as the order of the corresponding singular curve, i.e., the order of a general point $\mathbf{Q} = \Phi(s,t)$ on $h(s,t) = 0$.

There can be two situations for a singular factor h: if the order of h is finite, then $h(s,t) = 0$ corresponds to a one-dimensional singular branch on the parametric surface; otherwise, if the order of h is ∞, then the whole curve $h(s,t) = 0$ is mapped to a single isolated singular point on the surface, for example, the vertex of a cone.

Singular Factors of Infinite Orders. We first show how to determine whether the order of a singular factor is infinite, i.e., whether the factor corresponds to an isolated singular point.

Let $h(s,t)$ be a singular factor. Suppose that $\deg_s(h) > 0$ and let $h_0(t)$ be the leading coefficient of h with respect to the variable s. Then there exist polynomials $\alpha_i(s,t), r_i(s,t)$ with $\deg_s(r_i) < \deg_s(h)$, such that:

$$h_0(t)^\gamma f_i(s,t) = h(s,t)\alpha_i(s,t) + r_i(s,t),\ i = 0,1,2,3, \tag{10}$$

where γ is a nonnegative integer and r_i are not all zero (Chapter 2, Section 3 in Cox et al. [1998]). Then we have the following result.

Proposition 4.8. *$h(s,t) = 0$ is the preimage of an isolated singular point $\mathbf{Q}_0 = (x_0,y_0,z_0,w_0)$ if and only if*

$$r_0(s,t) : r_1(s,t) : r_2(s,t) : r_3(s,t) = x_0 : y_0 : z_0 : w_0 \tag{11}$$

for every (s,t), where x_0,y_0,z_0,w_0 are not all zero.

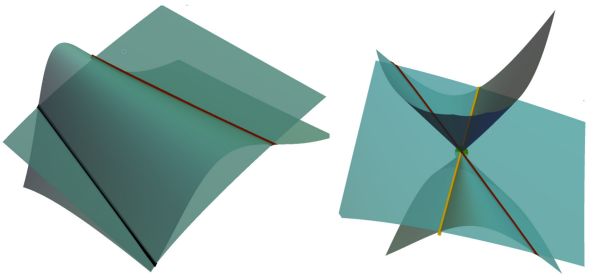
Proof. For the proof, please refer to Appendix A. □

Remark 2. If $\deg_s(h) = 0$, we can similarly take t as the prime variable and then Equation (10) becomes:

$$f_i(s,t) = h(t)\alpha_i(s,t) + r_i(s,t),\ i = 0,1,2,3,$$

where $\deg_t(r_i) < \deg_t(h)$. The statement of Proposition 4.8 still holds.

[1] https://github.com/casgeo/MovingPlane.



(a) (b)

Fig. 4. Singular points corresponding to singular factors in Example 4.2: (a) Local illustration of the self-intersection curves corresponding to $h_3 = 0$ (black curve) and $h_4 = 0$ (red curve); (b) Local illustration of the self-intersection curves corresponding to $h_2 = 0$ (yellow curve), $h_4 = 0$ (red curve) and the isolated singular point corresponding to $h_1 = 0$.

Example 4.2 (Continued). We continue to check each singular factor in Example 4.2. For $h_1 = s$, $(r_0 : r_1 : r_2 : r_3) = (1 : 0 : 0 : 1)$. Hence, $h_1 = s = 0$ is the preimage of an isolated singular point $\mathbf{Q}_0 = (1, 0, 0, 1)$. For $h_2 = t - 1$, we choose t as the prime variable and compute the four residues $(r_0 : r_1 : r_2 : r_3) = (1 : 2s(s-1) : 0 : 1)$, hence $h_2 = t - 1 = 0$ gives the preimage of a one-dimensional singular curve of a finite order. Similarly, both $h_3 = 0$ and $h_4 = 0$ are the pre-images of one-dimensional singular curves of finite order.

Singular Factors of Finite Orders. Next we cope with singular factors of finite orders based on LU decomposition of polynomial matrices. The reader is referred to Appendix B for details.

Theorem 4.9. *Let $h(s,t)$ be a singular factor of finite order with $\deg_t(h) \geq 1$ and let $\mathbb{N}(s,t)$ be the matrix of size $p \times k$ in (9). For a generic $s = s_0$, perform LU decomposition on the univariate polynomial matrix $\mathbb{N}(s_0,t)$ such that $\mathbb{N}(s_0,t) = PL(t)U(t)$, where P is a permutation matrix, $L(t)$ is a unitary lower triangular matrix and U is an upper triangular matrix (please refer to (14) in Appendix B for details). Let*

$$U_i(t) := \gcd(u_{i,i}, u_{i,i+1}, \ldots, u_{i,k}), \quad i = 1, \ldots, p. \tag{12}$$

Then

(1) $U_p(t) = 0$.
(2) Let α be the total number of $i \in \{1, \ldots, p-1\}$ such that $h(s_0,t) | U_i(t)$, then the order of $h(s,t)$ is $\alpha + 1$.

Proof. The proof is given in Appendix C. □

Example 4.2 (Continued). We now continue to compute the orders of the singular factors h_2, h_3, h_4. Select a random rational number $s_0 = \frac{2}{3}$, and compute the LU decomposition of $\mathbb{N}(s_0,t)$ and $\{U_i(t)\}_{i=1}^7$. Then we get $U_7(t) = (t-1)(6t^3 + 14t^2 - 34t - 9)$, $U_6(t) = t$ and $U_i(t) = 1, i = 1, \ldots, 5$. Since $h_2 = t - 1$ divides only U_7, h_2 is an order-two singular factor. Similarly, since $h_4(s_0,t) = \frac{8}{9}t^3 + \frac{56}{27}t^2 - \frac{136}{27}t - \frac{4}{3}$ divides only U_7, h_4 is also an order-two singular factor.

Similarly for $h_3 = s^2 + 4s - 4$, we can choose a random value $t = t_0$ and compute the LU decomposition of $\mathbb{N}(s,t_0)$. One can easily check that h_3 is of order two.

5 ALGORITHM

Our algorithm for computing singularities of rational surfaces contains two parts: (1) computation of singular factors; and (2) determination of the order of each singular factor.

5.1 Fast Computation of Singular Factors

The outline for computing the singularities of a rational parametric surface is first to compute a matrix representation $\mathbb{M}(X)$, which can be done by solving a linear system of Equations (5). After that, we compute the matrix $\mathbb{N}(s,t)$ in (9), and then compute the determinant factor $H(s,t)$ of order $p-1$, i.e., the GCD of all the $(p-1) \times (p-1)$ minors of the matrix $\mathbb{N}(s,t)$. Finally, we factor $H(s,t)$ into irreducible singular factors.

Note that when the size of the polynomial matrix $\mathbb{N}(s,t)$ is big, computing all the $(p-1) \times (p-1)$ minors can be time consuming. Here we adopt the KKS algorithm proposed in Kaltofen et al. [1987] to finish the task. We randomly generate matrices $U_1, U_2 \in \mathbb{R}^{(p-1)\times p}$ and $V_1, V_2 \in \mathbb{R}^{k\times(p-1)}$ and compute $g_1(s,t) = \det(U_1\mathbb{N}(s,t)V_1)$ and $g_2(s,t) = \det(U_2\mathbb{N}(s,t)V_2)$. Then $H(s,t) = \gcd(g_1(s,t), g_2(s,t))$ with probability one.

Algorithm 1 outlines the main approach for computing all the singular factors.

5.2 Order Computation of Singular Factors

We compute the order of each singular factor based on Proposition 4.8 and Theorem 4.9. The details are described in Algorithm 2. Note that in Step 3 of Algorithm 2, an LU decomposition of a univariate polynomial matrix $\mathbb{N}(s,t_0)$, or $\mathbb{N}(s_0,t)$ is needed. Since this matrix is independent of each factor h, the LU decomposition needs to be computed only once for different singular factors.

6 EXAMPLES AND DISCUSSION

In this section, we present various examples to compare our algorithm with existing methods from different perspectives including robustness, efficiency, order computation and numerical stability. The state-of-the-art methods we compare with include the resultant method [Busé et al. 2008], the Gröbner basis method [Huang and Wang 2011] and the characteristic set method [Pérez-Díaz 2007], which we abbreviate as BEG, HW, and PD algorithms. The overall performance of different methods are summarized in Table 1.

The BEG algorithm is invalid in the presence of certain base points. The BEG and HW algorithms do not compute cusp curves or isolated singular points, and sometimes either provide extraneous singular factors or miss true singular factors; moreover, they cannot compute orders of singular points, and the output equations are on parameters (s,u) but not (s,t) (see detailed explanation on this point in Example 6.1). The BEG, HW, and PD algorithms do not apply to numerical computations. In contrast, our algorithm does not have these defects and is comparatively numerically stable.

We illustrate the performance of these methods and our algorithm using simple surfaces, surfaces with isolated singular points of infinite orders, surfaces with base points in the parametrization, the classic Utah teapot patches, and so on. Note that in all the

ALGORITHM 1: Compute singular factors of a rational surface

Input : A rational parametric surface of bi-degree (d_1, d_2):
$\Phi(s, t) = (f_0(s, t), f_1(s, t), f_2(s, t), f_3(s, t))$.
Output: Singular factors $J = \{h_i(s, t)\}_{i=1}^{l}$ of $\Phi(s, t)$.
Steps :

(1) Set $v = (d_1 - 1, 2d_2 - 1)$ if $d_1 \geq d_2$; otherwise $v = (2d_1 - 1, d_2 - 1)$;
(2) Compute the matrix representation $\mathbb{M}(X)$ in (7), which has $p = 2d_1d_2$ rows;
(3) Substitute $X = \Phi(s, t)$ into $\mathbb{M}(X)$ to get the matrix $\mathbb{N}(s, t)$;
(4) Compute $H(s, t)$, the GCD of all $(p - 1) \times (p - 1)$ minors of $\mathbb{N}(s, t)$ using the KKS algorithm;
(5) Factor $H(s, t) = \prod_{i=1}^{l} h_i(s, t)^{l_i}$;
(6) Output h_i, $i = 1, \ldots, l$.

examples, if not specially mentioned, the parametrizations of the surfaces are proper, i.e., the surface is not multiply painted.

6.1 Examples

We compare the robustness and computational efficiency of different approaches for Examples 6.1–6.6 in Table 2, where the column '#W' refers to the number of incorrect singular factors in the result; for example, +3 means there are 3 extraneous singular factors which do not correspond to any singularities, and −1 means the result misses one true singular factor.

Example 6.1 (Simple Example). The first example is taken from Jia et al. [2009]. Consider a rational ruled surface

$$\Phi(s, t) = \Phi_0(s) + t\Phi_1(s)$$

of bi-degree $(2, 1)$ in (s, t) with

$$\Phi_0(s) = (s + 3, 1, s^2 - 3s + 1, s)$$

and

$$\Phi_1(s) = (1, s^2 + 1, 2s, s + 3).$$

Let $v = (d_1 - 1, 2d_2 - 1) = (1, 1)$. By Algorithm 1, we compute a 4×4 representation matrix $\mathbb{M}(X) = (M_1, M_2, M_3, M_4)$:

$$M_1^T = \begin{pmatrix} -10x + 25y + 5z \\ -5x + 40y \\ 58x - 58y - 131z \\ -152x - 40z + 138w \end{pmatrix}, M_2^T = \begin{pmatrix} -209x + 695y - 68z \\ 68x - 199y \\ -16x - 398y - 40z + 138wz \\ -58x + 199 \end{pmatrix},$$

$$M_3^T = \begin{pmatrix} 53x - 98y - 61z \\ -77x - 5y + 138w \\ 10x - 10y + 25z \\ -50x + 5z \end{pmatrix}, M_4^T = \begin{pmatrix} 52x - 199y + 43z + 138w \\ -43x + 68y \\ -136x + 136y + 5z \\ -10x - 68z \end{pmatrix}.$$

Substituting the parametrization $\Phi(s, t)$ into the matrix $\mathbb{M}(X)$ we get the matrix $\mathbb{N}(s, t)$. Computing the gcd of the 3×3 minors of $\mathbb{N}(s, t)$ we obtain

$$h_1 = 111s^2t^2 + 139s^2t + 483st^2 + 59s^2 - 353st + 631t^2 - 413s - 1561t + 879.$$

Checking the order of h_1 by Algorithm 2, we find that $h_1 = 0$ is the preimage of an order 2 self-intersection curve. The PD algorithm returns the same result. See Figure 5(a) for an illustration.

By the BEG or HW algorithm, the computed singular factor is given by the parameter pair (s, u) satisfying $\Phi(s, t) = \Phi(u, v)$ for $s \neq u$:

$$h(s, u) = -s^2u^2 - s^2u - su^2 - 3s^2 + 15su - 19u^2 - s + 27u - 5.$$

ALGORITHM 2: Compute the order of a singular factor $h(s, t)$.

Input: The matrix representation $\mathbb{N}(s, t)$ and a singular factor $h(s, t)$.
Output: The order r of $h(s, t)$.
Steps :

(1) If $\deg_s(h) > 0$, let $h_0(t)$ be the leading coefficient of h in the variable s; compute the residue $r_i(s, t)$ such that $h_0^{\gamma}(t)f_i(s, t) = h(s, t)\alpha_i(s, t) + r_i(s, t)$ for $i = 0, 1, 2, 3$, and $\deg_s(r_i) < \deg_s(h)$;

Else if $\deg_s(h) = 0$, compute the residue $r_i(s, t)$ such that $f_i(s, t) = h(t)\alpha_i(s, t) + r_i(s, t)$ for $i = 0, 1, 2, 3$ and $\deg_t(r_i) < \deg_t(h)$;

(2) If $r_0 : r_1 : r_2 : r_3 = x_0 : y_0 : z_0 : w_0$ for a point $\mathbf{Q}_0 = (x_0, y_0, z_0, w_0) \in \mathbb{PR}^3$, then $r = \infty$, and $h = 0$ gives the preimages of the isolated singular point $\mathbf{Q}_0$; otherwise go to step (3)–(5);

(3) If $\deg_s(h) > 0$, choose a generic number $t = t_0$ and compute the LU decomposition of $\mathbb{N}(s, t_0)$;

Else if $\deg_s(h) = 0$, choose a generic number $s = s_0$ and compute the LU decomposition of $\mathbb{N}(s_0, t)$;

(4) If $\deg_s(h) > 0$, compute $U_i(s)$ in (12) for $i = 1, \ldots, p$. Let α be the number of $i \in \{1, \ldots, p - 1\}$ such that $h(s, t_0)|U_i(s)$;

Else if $\deg_s(h) = 0$, compute $U_i(t)$ in (12) for $i = 1, \ldots, p$. Let α be the number of $i \in \{1, \ldots, p - 1\}$ such that $h(s_0, t)|U_i(t)$;

(5) The order of $h(s, t)$ is $\alpha + 1$.

Although this result corresponds to the same self-intersection curve C in 3D space as our result, it is not straightforward to generate points on C from $h(s, u) = 0$ since (s, u) is not a natural parameter pair corresponding to a 3D point. Further techniques to compute the parameter pair (s, t) from (s, u) are needed. (This issue is addressed in the last column in Table 1.) Moreover, their algorithms do not tell the exact order of the self-intersection curve. These defects of the BEG and HW algorithms are presented in all the follow-up examples, and we shall not repeat them.

Example 6.2 (Cusp Example without Base Point). Consider a rational ruled surface of bi-degree $(3, 1)$

$$\Phi(s, t) = (s^2 + t, t(s^2 + 1) + s^2 + 1, -s^3 + 2s^2t + s^2 + 1, t(s^3 + 3) + 1).$$

Let $(v_1, v_2) = (d_1 - 1, 2d_2 - 1) = (2, 1)$. The computed matrix representation $\mathbb{M}(X)$ is of size 6×6, so is the matrix $\mathbb{N}(s, t)$. Our algorithm produces two singular factors: $h_1 = s$, which gives a double cusp curve on the surface, and

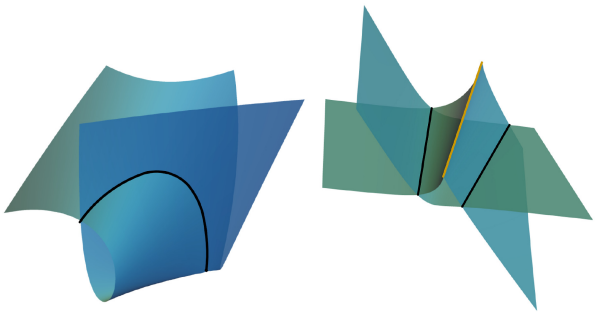
$$\begin{aligned} h_2 = {} & s^6t^4 - \frac{8}{5}s^6t^3 - \frac{2}{5}s^5t^4 - \frac{34}{5}s^6t^2 + \frac{1}{5}s^4t^4 + \frac{88}{5}s^6t \\ & - \frac{12}{5}s^5t^2 + \frac{26}{5}s^3t^4 - \frac{51}{5}s^6 + \frac{48}{5}s^5t + 10s^4t^2 \\ & - \frac{64}{5}s^3t^3 + \frac{18}{5}s^2t^4 - 10s^5 - \frac{112}{5}s^4t + \frac{8}{5}s^3t^2 \\ & - \frac{64}{5}s^2t^3 + \frac{89}{5}s^4 + \frac{16}{5}s^2t^2 + \frac{1}{5}t^4 + \frac{14}{5}s^3 \\ & + \frac{144}{5}s^2t + \frac{16}{5}t^3 - \frac{98}{5}s^2 - \frac{14}{5}t^2 - \frac{48}{5}t + \frac{33}{5}, \end{aligned}$$

which gives a double self-intersection curve. See Figure 5(b) for an illustration.

However, the HW algorithm gives only the result $h(s, u) = s^3u^2 - s^2u^2 - s^3 + 2s^2u + s^2 + u^2 - 3 = 0$, which

Table 1. Comparison of Different Methods in Terms of (1) Validity in the Presence of Base Points, (2) Cusps and Isolated Singular Points are Computed Correctly, (3) Extraneous Factors or Missing Factors are Produced, (4) Correct Orders of Singularities are given, (5) the Type of Equations in Defining the Singularities and (6) Whether or Not the Method Applicable to Numerical Computations

	Validity in base points	Cusps or isolated singularities	Extra factors or missing factors	Orders	Equations	Numerical stablility
BEG	N	N	Y	N	$h(s,u)$	N
HW	Y	N	Y	N	$h(s,u)$	N
PD	Y	Y	N	Y	$h(s,t)$	N
Ours	Y	Y	N	Y	$h(s,t)$	Y



(a) (b)

Fig. 5. (a) A double self-intersection curve(black) in Example 6.1; (b) A cusp curve and a self-intersection curve (the two thick black curves are two trimmed parts of one self-intersection curve) in Example 6.2.

essentially corresponds to $h_2(s,t) = 0$. Their result does not contain the cusp curve corresponding to $h_1 = 0$. The BEG algorithm generates two singular factors corresponding to our singular factors. The PD algorithm produces the same singular factors as ours, but this method takes much longer computational time.

Example 6.3 (Example with Base Points). Consider a bi-cubic surface

$$\Phi(s,t) = (-s^3(t^2-1), (s+2)st, -t^3(s^2-4), s^3),$$

which has a base point at $(s,t) = (0,0)$. The computed representation matrix $\mathbb{M}(X)$ is of size 18×26, so is the matrix $\mathbb{N}(s,t)$. By our algorithm, we can find three singular factors: $h_1 = s$, $h_2 = t$, and $h_3 = s+2$. $h_1 = 0$ and $h_2 = 0$ correspond to isolated singular points $\mathbf{Q}_1 = (1,0,0,1)$ and $\mathbf{Q}_2 = (0,0,1,0)$, and $h_3 = 0$ corresponds to a self-intersection curve. Figure 6(a) illustrates the surface with the self-intersection curve and the singular point $\mathbf{Q}_1$. Note that the other singular point $\mathbf{Q}_2$ is at infinity.

The BEG algorithm becomes invalid since the determinant of the resultant matrix vanishes. The HW algorithm gives only the singular factor corresponding to h_3 but misses the other two factors.

Example 6.4 (Utah Teapot). In this example, we compute the singularities for one of the 32 patches of the classic Utah teapot (a generic patch that is in no way special compared to the other patches) which is a bi-degree $(3,3)$ polynomial surface:

$$\begin{aligned}\Phi(s,t) = (&-256s^3t^3 + 1056s^3t^2 + 624s^2t^3 - 2574s^2t^2 - 240st^3\\ &- 800s^3 + 990st^2 + 1792t^3 + 1950s^2 - 7392t^2 - 750s + 5600,\\ &- 256s^3t^3 - 288s^3t^2 + 624s^2t^3 + 1344s^3t + 702s^2t^2 - 240st^3\\ &- 3276s^2t - 270st^2 + 1792t^3 + 1260st + 2016t^2 - 9408t,\\ &- 1575s^2 + 1575s + 9600, 4000).\end{aligned}$$

Our algorithm gives four singular factors: $h_1 = s^2 - s - \frac{1}{2}$, $h_2 = t^2 - t - \frac{25}{8}$, $h_3 = s^3 - \frac{39}{16}s^2 + \frac{15s}{16} - 7$, and h_4 is a bi-degree (6,6) polynomial. h_1, h_2, h_4 are all order 2 singular factors, and $h_3 = 0$ corresponds to three isolated singular points of infinite order. Figure 6(b) illustrates the singularities.

The HW algorithm produces three polynomials in s,u corresponding to h_2, h_3, h_4, but misses h_1. The BEG algorithm outputs $h_1h_2h_3^{10}H_1(s,u)H_2(u)$, where $H_1(s,u)$ is an extraneous factor of bi-degree $(10,6)$ and $H_2(u)$ is an extraneous factor of degree 6. BEG misses the factor $h_4 = 0$. Note that none of the singular factors have real branches within the region $[0,1]\times[0,1]$, which agrees with the fact that the Utah teapot patches are self-intersection free within $[0,1]\times[0,1]$.

Example 6.5. Consider a bi-degree $(4,2)$ surface defined by

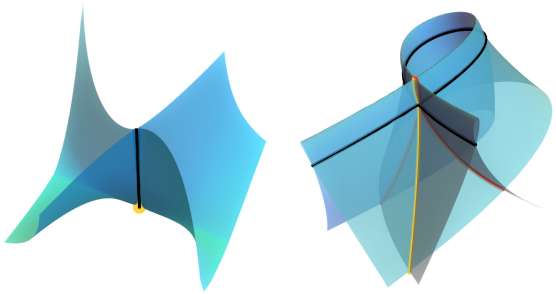
$$\begin{aligned}\Phi(s,t) = (&4s^3 + st^2 + 4s^2 - 12st + t^2 + s + 1,\\ &4s^4 + s^2t^2 + s^2 + 6t, 6t^2, 4s^2 + t^2 + 1).\end{aligned}$$

Let $(\nu_1,\nu_2) = (d_1-1, 2d_2-1) = (3,3)$. The representation matrix $\mathbb{M}(X)$ and $\mathbb{N}(s,t)$ are both of size 16×25. By our algorithm, we are able to compute three self-intersection curves of order two, whose pre-images are given by $h_1 = 4s^2+t^2+1 = 0$, $h_2 = 4s^2+t^2-12t+1 = 0$ and $h_3 = 16s^4 + 4s^2t^2 + 8s^2 + t^2 - 12t + 1 = 0$. Note that $h_1 = 0$ corresponds to a singular curve in 3D complex space. Figure 7(a) depicts the two real self-intersection curves on this surface.

The BEG algorithm gives $u^4(4s^2+1)^4(4s^2+u^2-12u+1)(16s^4 + 4s^2u^2 + 8s^2 + u^2 - 12u + 1)^2$, which has extraneous factors u^4 and $4s^2+1$ but loses the factor h_1. The HW algorithm returns only the singular factor $4s^2 + u^2 - 12u + 1 = 0$, but loses h_1 and h_3.

Notice that $\Phi(s,t)$ can also be regarded as a triangular surface of total degree 4. If we apply our algorithm for triangular surfaces, then we get a matrix representation $\bar{\mathbb{M}}(X)$ of size 28×55 which is much larger than that of $\mathbb{M}(X)$. Of course, our algorithm finally produces the same result for both representations.

Remark 3. Any parametric surface $\Phi(s,t)$ can be taken as either a tensor product surface of bi-degree (d_1,d_2) or a triangular surface of total degree d with a finite number of base points, regardless of the type of the Newton polygon. Thus our algorithm is impervious to base points based on Proposition 4.4. The matrix representations



(a) (b)

Fig. 6. (a) A self-intersection curve (black) and an isolated singular point (yellow) in Example 6.3; (b) Three double self-intersection curves (in black, red and yellow) in Example 6.4.

for the two types of surfaces have $2d_1d_2$ rows and $d(2d-1)$ rows respectively. Since $d \geq \max(d_1, d_2)$, unless $d = \max(d_1, d_2)$, the tensor-product form has a matrix representation of smaller size than that of the triangular form.

Example 6.6. Consider an example from Jia et al. [2009] which has singularities of higher order. Let

$$\Phi(s,t) = \Phi_0(s) + t\Phi_1(s),$$

where

$$\Phi_0(s) = (7s^7 + 77s^6 - 315s^5 + 385s^4 - 175s^3 + 21s^2, 49s^7 - 84s^6 + 21s^5 + 35s^3 - 21s^2, 56s^7 - 147s^6 + 126s^5 - 35s^4, 1)$$
$$\Phi_1(s) = (1,1,1,0).$$

Let $(\nu_1, \nu_2) = (6,1)$. Then the matrices $\mathbb{M}(X)$ and $\mathbb{N}(s,t)$ are both of size 13×21. Our algorithm outputs three singular factors: $h_1 = s$, $h_2 = s-1$, $h_3 = t$, and $h_4(s,t)$ which is a degree 22 irreducible polynomial in s. By Algorithm 2, both $h_1 = 0$ (with multiplicity 2) and $h_2 = 0$ give an order three self-intersection curve (actually these factors correspond to the same self-intersection curve in 3D space), $h_3 = 0$ gives an isolated singular point $\mathbf{Q} = (1,1,1,0)$ at infinity, and $h_4 = 0$ gives a double self-intersection curve. Figure 7(b) illustrates the double and the triple self-intersection curves.

The BEG algorithm returns no result on this example since the determinant of their resultant vanishes. The HW algorithm misses the locus $h_3 = 0$, and runs over 70 seconds; moreover, it cannot tell the different orders of the self-intersection curves. The PD algorithm returns the same result as ours.

6.2 Discussion

In this section, we discuss the pros and cons of the four mentioned algorithms from different perspectives including robustness, efficiency, order computation, and numerical stability.

Robustness. Considering the robustness of the above-mentioned algorithms, our algorithm and the PD algorithm always return correct singular factors. In contrast, the BEG algorithm usually produces some extraneous factors that do not correspond to any singularities. Occasionally, the BEG algorithm also misses certain singular factors. The HW algorithm often misses some singular factors. Those missed singular factors by the BEG and the HW algorithms either correspond to cusp curves or isolated singular points of infinite order. Actually, this phenomena has been addressed in Huang and Wang [2011], where the authors clarify that their computation system is built on $\Phi(s,t) = \Phi(u,v)$ for $s \neq u$, which is also used in the BEG algorithm. Thus, both the BEG and the HW algorithms return only singular factors that correspond to self-intersection curves.

On the other hand, both the HW and the BEG algorithms return polynomial factors $h(s,u)$ corresponding to self-intersection curves of a surface. Since (s,u) is not a natural parameter pair for a singular point, further computations are needed to transfer from (s,u) to (s,t) or (u,v). On comparison, our method and the PD method both directly return singular factors $h(s,t)$; hence, singular curves in 3D space can be computed directly by mapping $h(s,t) = 0$ through Φ.

Besides, in some situations the BEG algorithm becomes invalid since the determinant of the resultant matrix vanishes, hence does not return any result. This degeneracy usually occurs when the parametrization of the surface contains base points. However, our algorithm is impervious to base points. Therefore, it works well for any type of rational parametrization, because parametrizations with any type of Newton polygons can be taken as tensor-product surfaces with a finite number of base points.

Efficiency. We implemented the four approaches in Maplesoft 2018 on an Inter(R)Core(TM)i7-7700K CPU @ 4.20GHz with RAM 32G. The comparison of the computation time for Examples 6.1-6.6 by the four approaches is shown in Table 2. It can be seen that both the BEG method and our method are efficient, while the HW and the PD algorithms report longer computation times.

In our implementation, when the surface has no base points, we use the Dixon resultant matrix as the representation matrix to accelerate the algorithm, instead of solving the linear system (5). To more fairly compare the efficiency of the four methods, we generate 20 random examples for each bi-degree parametric surface that contains full term monomials in given degree with random rational coefficients in $[-5,5]$. Note that these examples do not have any base points. We compute the average computation time for each bi-degree which is shown in Table 3. In summary, the BEG algorithm and our method are most efficient and have comparable implementation time. Note that, in some examples, the computation time of our algorithm is a little bit slower than that of the BEG algorithm, largely because we are computing $h(s,t) = 0$ while the BEG algorithm is computing $g(s,u) = 0$, and h usually has a higher degree than g. As we have addressed previously, $h(s,t)$ is preferable to $g(s,u)$. The HW and the PD algorithms are computationally very expensive for relatively high degree surfaces since computation of Gröbner bases or regular systems is involved.

Order Computation. Both our algorithm and the PD algorithm can compute the exact order of each singular factor, while the BEG and the HW algorithms cannot compute the orders of singular factors within their own framework. This reason is also why the BEG method cannot get rid of extraneous factors. Note that our method can get all the correct singular factors at the first stage (Algorithm 1) by computing, if needed, the determinant factor of order $p-1$. Even if we only compute one minor which may contain extraneous factors at the first stage, we can still get rid of the

Table 2. Robustness and Computation Time (in Seconds) of Different Approaches

		BEG		HW		PD		Ours	
	degree	Time	#W	Time	#W	Time	#W	Time	#W
Eg 6.1	(2,1)	0.047	0	0.250	0	0.328	0	0.015	0
Eg 6.2	(3,1)	0.078	0	4.391	−1	171.531	0	0.047	0
Eg 6.3	(3,3)	–	–	0.078	−2	1.610	0	0.187	0
Eg 6.4	(3,3)	0.281	+1	0.375	−1	6.828	0	0.438	0
Eg 6.5	(4,2)	0.188	+2 &−1	0.063	−2	4.359	0	0.265	0
Eg 6.6	(7,1)	–	–	76.828	−1	0.750	0	0.500	0

#W refers to the number of extraneous or missing singular factors, "+n" means n extraneous singular factors are obtained, "-n" means n singular factors are missed, "0" means correct singular factors are generated. "–" means no results are returned.

Table 3. Average Computation Time (in Seconds) of 20 Full-term Parametrization Examples for Different Approaches

Degree	BEG	HW	PD	Ours
(2,1)	0.062	1.112	0.964	0.016
(3,1)	0.080	110.600	650.324	0.031
(4,1)	0.202	1374.600	>5000.000	0.157
(2,2)	0.132	406.484	569.121	0.109
(3,2)	0.391	>5000.000	>5000.000	0.578
(4,2)	1.562	>5000.000	>5000.000	3.201

(a) (b)

Fig. 7. (a) Two double self-intersection curves (in black and yellow) in Example 6.5; (b) A double self-intersection curve (in black) and a triple self-intersection curve (in yellow) in Example 6.6.

extraneous factors at the second stage by computing the order of each singular factor. Hence, order computation is not only an advantage of our algorithm but also greatly accelerates our algorithm.

Improper Parametrization. Previously, we have assumed that the surface parametrization is proper. However, improper parametrization is very common in practice. If the surface parametrization is improper, the BEG algorithm and the HW algorithm become invalid, while our algorithm and the PD algorithm still work well. We illustrate with an example.

Example 6.7. Consider an improperly parameterized surface

$$\Phi(s,t) = (-s^3(t^2-1), (s+2)st, -t^3(s^2-4), s^3).$$

Both the BEG and the HW algorithms fail since zeros are returned during the computation. Our algorithm gives three singular factors: $h_1 = s, h_2 = s+2$, and $h_3 = t$, where $h_1 = 0$ and $h_3 = 0$ correspond to two isolated singular points $\mathbf{Q}_1 = (0,0,1,0)$ and $\mathbf{Q}_2 = (1,0,0,1)$, and the singular factor $h_2 = 0$ corresponds to an order four self-intersection curve. The PD algorithm returns the same results.

Numerical Stability. Our algorithm works also for parametric surfaces with floating point coefficients. Note that in such situations, an approximate GCD algorithm and an approximate factorization algorithm of bivariate polynomials with floating point coefficients are needed. In this article, we employ the approximate GCD and factorization algorithms for multivariate polynomials provided by Kaltofen et al. [2006] and Zeng [2008].

According to a large number of experiments, our algorithm manifests numerical stability. That is, if the input data contains some numerical error ε, then the output data has an error bounded by $K\varepsilon$, where K is a positive constant that depends on the algorithm. However, we are not able to provide a theoretical proof about the numerical stability of our algorithm since complex polynomial GCD and factorization are involved. In the following, we give two examples to illustrate the fact.

Example 6.1 (Continued). We recompute Example 6.1 using floating point arithmetic with double precision. The approximate GCD of all the 3×3 minors of $\mathbb{N}(s,t)$ is

$$\begin{aligned}\bar{h}_1(s,t) = &-.3024343413303094s^2t^2 - .3787240850893085s^2t\\ &- 1.315998079842698st^2 - .1607533886350279s^2 + .9617956981044943st\\ &- 1.719243868283114t^2 + 1.125273720445204s + 4.253153214564076t\\ &- 2.394953027291350.\end{aligned}$$

By the approximate GCD factorization algorithm proposed in Kaltofen et al. [2006], $\bar{h}_1(s,t)$ can not be factored. Now we compare the difference between $h_1(s,t)$ and $\bar{h}_1(s,t)$ over a parametric domain, say $[0,1]\times[0,1]$. Write h_1 in Example 6.1 and $\bar{h}_1$ in their

Bernstein-Bézier forms:

$$h_1(s,t) = \sum_{i=0}^{2}\sum_{j=0}^{2} \alpha_{ij} B_i^2(s) B_j^2(t), \quad \bar{h}_1 = \sum_{i=0}^{2}\sum_{j=0}^{2} \bar{\alpha}_{ij} B_i^2(s) B_j^2(t)$$

whose coefficient matrices are

$$\alpha = \begin{pmatrix} \frac{293}{37} & \frac{197}{222} & -\frac{17}{37} \\ \frac{1345}{222} & -\frac{785}{444} & -\frac{385}{222} \\ \frac{175}{37} & -\frac{725}{222} & -\frac{25}{111} \end{pmatrix},$$

$$\bar{\alpha} = \begin{pmatrix} 7.918918918918855 & 0.8873873873873300 & -.4594594594594980 \\ 6.058558558558497 & -1.768018018018070 & -1.734234234234264 \\ 4.729729729729666 & -3.265765765765814 & -.2252252252252453 \end{pmatrix}.$$

Since $\max_{0\le i,j\le 2} |\alpha_{ij} - \bar{\alpha}_{ij}| \approx 10^{-14} \le K\varepsilon$ with $\varepsilon \approx 10^{-16}$ and $K \approx 100$, our algorithm manifests numerical stability.

Example 6.2 (Continued). We perform floating point arithmetic with double precision on Example 6.2. Computing the approximate GCD $H(s,t) = 0$ of all the minors of $\mathbb{N}(s,t)$ by the approximate GCD algorithm and factoring $H(s,t)$ with the approximate factorization algorithm, we obtain two factors $\bar{h}_1(s) = 1.000000000s$ and a bidegree $(6,4)$ factor

$$\begin{aligned}
&\bar{h}_2(s,t) = \\
&1.000000000000000s^6t^4 - 1.600000000000025s^6t^3 - .400000000000504s^5t^4 \\
&- 6.799999999999999s^6t^2 + 5.021279428 \times 10^{-14}s^5t^3 + .199999999999915s^4t^4 \\
&+ 17.600000000000015s^6t - 2.400000000000054s^5t^2 + 4.763450957 \times 10^{-14}s^4t^3 \\
&+ 5.200000000000102s^3t^4 - 10.200000000000015s^6 + 9.600000000000110s^5t \\
&+ 10.000000000000009s^4t^2 - 12.800000000000017s^3t^3 + 3.600000000000004s^2t^4 \\
&- 10.000000000000010s^5 - 22.400000000000016s^4t + 1.600000000000021s^3t^2 \\
&- 12.800000000000016s^2t^3 - 6.177807677 \times 10^{-14}st^4 + 17.800000000000017s^4 \\
&- 3.491867976 \times 10^{-14}s^3t + 3.200000000000066s^2t^2 + 0.199999999999979t^4 \\
&+ 2.799999999999961s^3 + 28.800000000000025s^2t + 3.708803871 \times 10^{-14}st^2 \\
&+ 3.200000000000030t^3 - 19.600000000000009s^2 - 2.799999999999994t^2 \\
&+ 4.039347693 10^{-14}s - 9.600000000000034t + 6.599999999999966.
\end{aligned}$$

Clearly, $h_1 = \bar{h}_1$. Moreover, write h_2 in Example 6.2 in its Bernstein-Bézier form:

$$h_1(s,t) = \sum_{i=0}^{6}\sum_{j=0}^{4} \alpha_{ij} B_i^6(s) B_j^4(t),$$

where

$$\alpha = \begin{pmatrix} \frac{33}{2} & \frac{21}{5} & \frac{4}{3} & -\frac{6}{5} \\ \frac{33}{5} & \frac{21}{5} & \frac{4}{3} & -\frac{6}{5} \\ \frac{6}{5} & \frac{6}{5} & \frac{6}{5} & \frac{6}{5} \\ \frac{397}{75} & \frac{253}{75} & \frac{46}{45} & -\frac{88}{75} \\ \frac{141}{50} & \frac{93}{50} & \frac{83}{150} & -\frac{11}{10} \\ \frac{38}{75} & \frac{46}{75} & \frac{142}{225} & -\frac{14}{25} \\ -\frac{4}{5} & \frac{2}{15} & \frac{71}{45} & \frac{2}{5} \end{pmatrix}.$$

Similarly, write $\bar{h}_2$ as

$$\bar{h}_2 = \sum_{i=0}^{6}\sum_{j=0}^{4} \bar{\alpha}_{ij} B_i^6(s) B_j^4(t),$$

where $\bar{\alpha} = [\bar{\alpha}_1, \bar{\alpha}_2]_{6\times 4}$ with

$$\bar{\alpha}_1 = \begin{pmatrix} 6.599999999999966 & 4.199999999999958 \\ 6.599999999999973 & 4.199999999999964 \\ 5.293333333333070 & 3.373333333333302 \\ 2.819999999999966 & 1.859999999999970 \\ .5066666666666272 & .6133333333333059 \\ -.8000000000000414 & .1333333333333125 \end{pmatrix},$$

$$\bar{\alpha}_2 = \begin{pmatrix} 1.333333333333283 & -1.200000000000049 \\ 1.333333333333291 & -1.200000000000039 \\ 1.022222222222191 & -1.173333333333357 \\ .5533333333333132 & -1.100000000000006 \\ .6311111111111070 & -.5599999999999858 \\ 1.577777777777796 & .6000000000000418 \end{pmatrix}.$$

Since $\max |\alpha_{ij} - \bar{\alpha}_{ij}| \approx 10^{-14}$ for $0 \le i \le 6, 0 \le j \le 4$, our algorithm demonstrates numerical stability again.

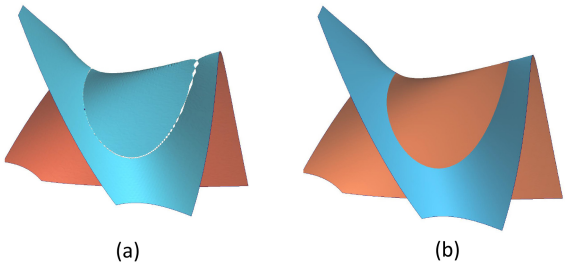


Fig. 8. The rendering result without considering the singular curve (a) and with special treatment of the singular curve (b).

Overall Comparison. In summary, our algorithm has the best performance considering all the above factors. The BEG algorithm is generally the most efficient algorithm, but BEG may produce many extraneous factors and BEG cannot tell the orders of the singular factors. Furthermore, BEG easily becomes invalid. The PD algorithm is the only algorithm that can ensure the same correct results as our algorithm; however, PD is very inefficient and impractical compared with other methods. The behavior of the HW algorithm is between the BEG algorithm and the PD algorithm: PD is efficient for generally low-degree surfaces but the computation costs increase rapidly as the degree of the parametric surface rises. Furthermore, HW sometimes misses singular factors and cannot tell the order of the singularities. When it comes to numerical computation environments, only our algorithm is valid.

7 APPLICATIONS IN RENDERING, MESHING, AND SURFACE INTERSECTIONS

Singularities are important features of surfaces and have substantial influence on subsequent processing of surfaces such as rendering, meshing, and surface/surface intersection.

Ray tracing in surfaces tends to generate aliasing effects in the neighborhood of singular curves. Figure 8(a) shows that the area around the singular curve has artifacts. Figure 8(b) shows a better rendering quality by using our computational results to locate in advance the singular curve.

Mesh generation of parametric surfaces often comes with big difficulties in regions around singularities, where a couple of common pitfalls of finite element modeling arise. Figure 9(a) shows the meshing result near the self-intersection curves where many triangles penetrate each other. This interpenetration might cause problems in later processing. Figure 9(b) shows the remeshing result of the parametric surface by specifically moving certain vertices of the triangles near the self-intersection curves onto the self-intersection curves, and hence the triangles no longer penetrate each other.

Singularities on surfaces also affect surface intersection. Tracing the intersection curve of two surfaces tends to fail near the singular points on one surface. An example is illustrated in Figure 10(a), where one of the two surfaces has a self-intersection curve. If we compute the intersection curve of the two surfaces by the tracing method, the intersection method misses two of the four branches near the singular curve (black) of one surface, as shown in Figure 10(b). Figure 10(c) improves the result by using

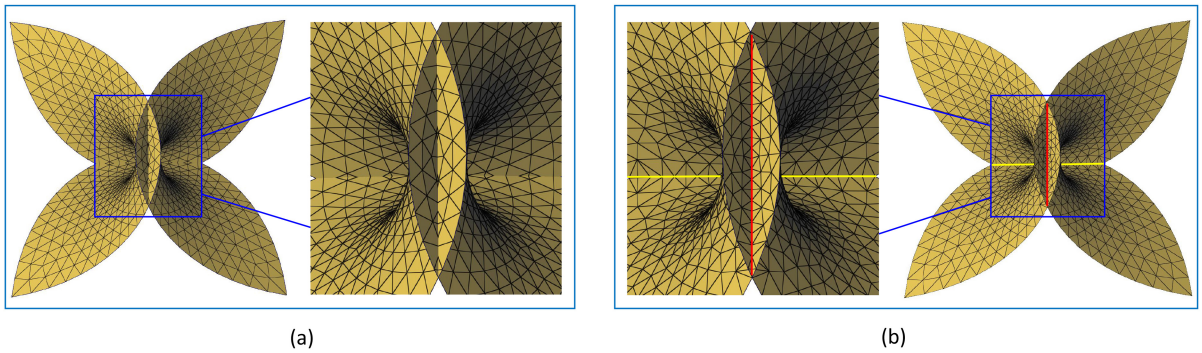


Fig. 9. (a) A triangular mesh of the Enneper surface without considering the singular curve, where many triangles near the self-intersection curves penetrate each other. (b) The remeshing result by moving certain vertices of the triangles near the self-intersection curves onto these curves, and hence the triangles no longer penetrate each other.

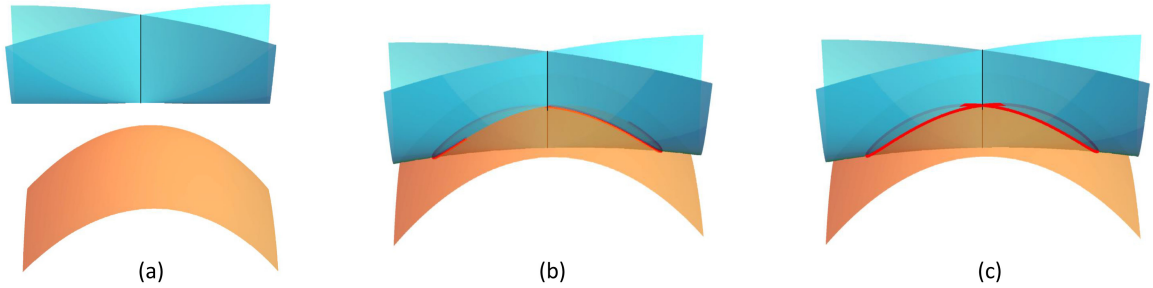


Fig. 10. Computing the intersection curve of two rational surfaces by the tracing method. (a) Two rational surfaces, one of which has a self-intersection curve; (b) The intersection curve (red) of the two surfaces is missing two of the four branches near the singular curve (black) of one surface; (c) the four branches of the intersection curve are all correctly generated by special treatment for the singular curve (black) on one surface.

special treatment for the singular curve (black) on one surface, and hence the four branches are all correctly generated.

8 CONCLUSIONS AND FUTURE WORK

In this article, we provide a symbolic algorithm for computing the singularities of rational surfaces. The algorithm provides the pre-images of all the singularities together with their orders. The algorithm is efficient, robust, and numerically stable. Applications of our algorithm in surface rendering, surface meshing, and surface/surface intersections demonstrate that singularity computation is essential for these applications.

As for future directions, the size of the representation matrix in the current work can possibly be reduced by replacing some moving planes with moving quadrics [Buse and Chen 2021; Lai et al. 2019]. Decreasing the size of the representation matrix can further improve the efficiency of our algorithm. Another future direction could be the direct factorization of the matrix $\mathbb{N}(s,t)$ into a form that can not only read off the singular factors but also their orders. Applications of singularities in other applications could also be explored.

APPENDICES

A PROOF OF PROPOSITION 4.8

Proof. "⇒": By assumption, for any (s^*,t^*) on $h(s,t) = 0$, $\Phi(s^*,t^*) = \mathbf{Q}_0$, i.e.,

$$f_0(s^*,t^*) : f_1(s^*,t^*) : f_2(s^*,t^*) : f_3(s^*,t^*) = x_0 : y_0 : z_0 : w_0.$$

Substituting $s = s^*, t = t^*$ into Equation (10) yields

$$h_0(t^*)^\gamma f_i(s^*,t^*) = r_i(s^*,t^*) \text{ for } i = 0,1,2,3.$$

Thus if $h_0(t^*) \neq 0$, then

$$r_0(s^*,t^*) : r_1(s^*,t^*) : r_2(s^*,t^*) : r_3(s^*,t^*) = x_0 : y_0 : z_0 : w_0.$$

Hence every parameter pair (s^*,t^*) satisfying $h(s,t) = 0$ and $h_0(t^*) \neq 0$ is a common solution of

$$\begin{aligned} F_1(s,t) &:= x_0 r_3 - w_0 r_0, \\ F_2(s,t) &:= y_0 r_3 - w_0 r_1, \\ F_3(s,t) &:= z_0 r_3 - w_0 r_2. \end{aligned} \tag{13}$$

That means

$$h(s,t)|F_1(s,t), h(s,t)|F_2(s,t), h(s,t)|F_3(s,t).$$

But since $\deg_s(F_i) < \deg_s(h)$, it follows that

$$F_1(s,t) \equiv 0, F_2(s,t) \equiv 0, F_3(s,t) \equiv 0.$$

Therefore,

$$r_0(s,t) : r_1(s,t) : r_2(s,t) : r_3(s,t) = x_0 : y_0 : z_0 : w_0.$$

"⇐" Let (s^*,t^*) be a point on $h = 0$. Then $h_0(t^*) \neq 0$; otherwise, Equations (10) and (11) together yield $x_0 = y_0 = z_0 = w_0 = 0$, which contradicts the assumption. Hence, Equation (10) gives

$$\begin{aligned} &f_0(s^*,t^*) : f_1(s^*,t^*) : f_2(s^*,t^*) : f_3(s^*,t^*) \\ =&r_0(s^*,t^*) : r_1(s^*,t^*) : r_2(s^*,t^*) : r_3(s^*,t^*) \\ =&x_0 : y_0 : z_0 : w_0 = \mathbf{Q}_0. \end{aligned}$$

This completes the proof. □

B LU DECOMPOSITION OF A POLYNOMIAL MATRIX

LU decomposition of a $p\times k(p \le k)$ matrix M factors M into a $p\times p$ lower triangular matrix L and a $p \times k$ upper triangular matrix U such that $M = LU$.

For a $p \times k(p \le k)$ polynomial matrix $M(t)$ with entries polynomials in t, a similar LU factorization $M(t) = PL(t)U(t)$ exists, where P is a constant permutation matrix, L is a unitary lower triangular matrix of size $p\times p$ and U is an upper triangular matrix of size $p\times k$. The entries of L and U are rational functions in t, and L has ones along its diagonal:

$$L = \begin{pmatrix} 1 & & & \\ \frac{l_{21}(t)}{d_1(t)} & 1 & & \\ \vdots & \ddots & \ddots & \\ \frac{l_{p,1}(t)}{d_1(t)} & \cdots & \frac{l_{p,p-1}(t)}{d_{p-1}(t)} & 1 \end{pmatrix},$$

and

$$U = \begin{pmatrix} \frac{u_{11}(t)}{v_1(t)} & \cdots & \cdots & \cdots & \frac{u_{1,k}(t)}{v_1(t)} \\ & \ddots & & & \vdots \\ & & \frac{u_{p,p}(t)}{v_p(t)} & \cdots & \frac{u_{p,k}(t)}{v_p(t)} \end{pmatrix}. \tag{14}$$

Obviously, $\mathrm{rank}(M) = \mathrm{rank}(U)$ for a generic value of t that is not a root of any $d_i(t) = 0, i = 1,\ldots,p-1$ or $v_i(t) = 0, i = 1,\ldots,p$. See Murota [1983] for more details.

C PROOF OF THEOREM 4.9

Proof. Since $\mathrm{rank}(\mathbb{N}(s,t)) < p$ for an arbitrary parameter pair (s,t) and $\mathrm{rank}(\mathbb{N}(s_0,t)) = \mathrm{rank}(U(t))$ for any generic value of t, we directly get $U_p(t) = 0$.

Let $t = t_0$ be a solution of $h(s_0,t) = 0$. Since $s = s_0$ is a generic value, by Proposition 4.3, the order of the point $\Phi(s_0,t_0)$ is $k = \mathrm{corank}(\mathbb{N}(s_0,t_0)) = \mathrm{corank}(U(t_0)) = \alpha + 1$. □

ACKNOWLEDGMENTS

The authors thank Prof. Yifei Chen from Institute of Mathematics, Chinese Academy of Sciences for his kind help on providing the important note on isolated singularities in algebraic geometry. The authors are also grateful to the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

D. Baraff, A. Witkin, and M. Kass. 2003. Untangling cloth. *ACM Transactions on Graphics* 22, 3 (2003), 862–870.
R. Barringer, C. J. Gribel, and Tomas A.-M. 2012. High-quality curve rendering using line sampled visibility. *ACM Transactions on Graphics* 31, 6 (2012), 1–10.
N. Botbol, L. Busé, and M. Chardin. 2014. Fitting ideals and mutiple points of surface parametrizations. *Journal of Algebra* 420 (2014), 486–508.
L. Busé. 2014. Implicit matrix representation of rational Bézier curves and surfaces. *C*omputer Aided Design 46 (2014), 14–24.
L. Buse and F. Chen. 2021. Determinantal tensor product surfaces and the method of moving quadrics. *Trans. Amer. Math. Soc.* 374 (2021), 4931–4952.
L. Busé and C. D'Andrea. 2012. Singular factors of rational plane curves. *J*ournal of Algebra 357 (2012), 322–346.
L. Busé, M. Elkadi, and A. Galligo. 2008. Intersection and self-intersection of surfaces by means of Bezoutian matrices. *C*omputer Aided Geometric Design 25 (2008), 53–68.
L. Busé, R. Goldman, and H. Schenck. 2016. Computational algebra and geometric modeling. *P*reface of Conference of Computational Algebra and Geometric Modeling (2016).
F. Chen, D. Cox, and Y. Liu. 2005. The μ-basis and implicitization of a rational parametric surface. *J*ournal of Symbolic Computation 39 (2005), 689–706.
F. Chen, W. Wang, and Y. Liu. 2008. Computing singular points of plane rational curves. *J*ournal of Symbolic Computation 43, 2 (2008), 92–117.
D. A. Cox, J. Little, and D. O'Shea. 1998. *I*deals, Varieties and Algorithms. Springer.
B. R. de Araújo, D. S. Lopes, P. Jepp, J. A. Jorge, and B. Wyvill. 2015. A survey on implicit surface polygonization. *Comput. Surveys* 47, 4, article 60 (2015).
G. Elber, T. Grandine, and M.-S. Kim. 2009. Surface self-intersection computation via algebraic decomposition. *C*omputer-Aided Design 41, 12 (2009), 1060–1066.
A. Gueziec, G. Taubin, F. Lazarus, and B. Hom. 2001. Cutting and stitching: Converting sets of polygons to manifold surfaces. *I*EEE Transactions on Visualization and Computer Graphics 7, 2 (2001), 136–151.
D. Harbinson, R. Balsys, and K. Suffern. 2019. Hybrid polygon-point rendering of singular and non-manifold implicit surfaces. In *2*019 23rd International Conference in Information Visualization Part II. 160–166.
R. Hartshorne. 1977. *A*lgebraic Geometry. Springer-Verlag.
J. Hoschek, D. Lasser, and L. L. Schumaker. 1993. *F*undamentals of Computer Aided Geometric Design. A. K. Peters, Ltd.
Y. Huang and D. Wang. 2011. Computing intersection and self-intersection loci of parametrized surfaces using regular systems and Gröbner bases. *C*omputer Aided Geometric Design 28 (2011), 566–581.
X. Jia, F. Chen, and J. Deng. 2009. Computing self-intersection curves of rational ruled surfaces. *C*omputer Aided Geometric Design 26 (2009), 287–299.
X. Jia and R. Goldman. 2009. μ-bases and singularities of rational planar curves. *C*omputer Aided Geometric Design 26 (2009), 970–988.
X. Jia and R. Goldman. 2012. Using Smith normal forms and μ-bases to compute all the singularities of rational planar curves. *C*omputer Aided Geometric Design 29 (2012), 296–314.
E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. 1987. Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM J. Algebraic Discrete Methods* (1987).
E. Kaltofen, Z. Yang, and L. Zhi. 2006. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *ISSAC* MMVI Proc. 2006 Internat. Symp. Symbolic Algebraic Comput., Jean-Guillaume Dumas (Ed.). ACM, New York, 169–176.
S. Krishnan and D. Manocha. 1997. An efficient surface intersection algorithm based on lower-dimensional formulation. *A*CM Transactions on Graphics 16, 1 (1997), 74–106.
Y. Lai, F. Chen, and X. Shi. 2019. Implicitizing rational surfaces without base points by moving planes and moving quadrics. *C*omputer Aided Geometric Design 70 (2019), 1–15.
Y. Li and J. Barbič. 2018. Immersion of self-intersecting solids and surfaces. *A*CM Transactions on Graphics 37, 4 (2018), 1–14.
Y.-M. Li and R. J. Cripps. 1997. Identification of inflection points and cusps on rational curves. *C*omputer Aided Geometric Design 14, 5 (1997), 491–497.
H. Lin, Y. Qin, H. Liao, and Y. Xiong. 2014. Affine arithmetic-based B-spline surface intersection with GPU acceleration. *I*EEE Transactions on Visualization and Computer Graphics 20, 2 (2014), 172–81.
C. Loop and J. Blinn. 2006. Real-time GPU rendering of piecewise algebraic surfaces. *A*CM Transactions on Graphics 25, 3 (2006), 664–670.
D. Manocha and J. F. Canny. 1992. Detecting cusps and inflection points in curves. *C*omputer Aided Geometric Design 9, 1 (1992), 1–24.
K. Murota. 1983. LU-decomposition of a matrix with entries of different kinds. *Linear Algebra Appl.* 49 (1983), 275–283.
D. Pekerman, G. Elber, and M.-S. Kim. 2008. Self-intersection detection and elimination in freeform curves and surfaces. *C*omputer-Aided Design 40, 2 (2008), 150–159.
S. Pérez-Díaz. 2007. Computation of the singularities of parametric plane curves. *J*ournal of Symbolic Computation 42, 8 (2007), 835–857.
S. Pérez-Díaz, J. R. Sendra, and C. Villarino. 2015. Computing the singularities of rational surfaces. *Math. Comp.* 84, 294 (2015), 1991–2021.
O. J. Peterson. 1917. The double points of rational curves. *T*he American Mathematical Monthly 24, 8 (1917), 376–379.
S. Plantinga and G. Vegter. 2004. Isotopic approximation of implicit curves and surfaces. In *2*004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing *(*SGP'04). Association for Computing Machinery, New York, 245–254.
M. Sakai. 1999. Inflection points and singularities on planar rational cubic curve segments. *C*omputer Aided Geometric Design 16, 3 (1999), 149–156.
T. Sakkalis and R. Farouki. 1990. Singular points of algebraic curves. *J*ournal of Symbolic Computation 9, 4 (1990), 405–421.
T. W. Sederberg and F. Chen. 1995. Implicitization using moving curves and surfaces. *A*CM SIGGRAPH 15 (1995).

T. W. Sederberg, R. Goldman, and H. Du. 1997. Implicitizing rational curves by the method of moving algebraic curves. *J*ournal of Symbolic Computation 23 (1997), 153–175.

L. Shen and R. Goldman. 2017. Implicitizing rational tensor product surfaces using the resultant of three moving planes. *A*CM Transactions on Graphics 36, 5, article 167 (2017).

X. Shi and F. Chen. 2010. Computing the singularities of rational space curves. In *I*nternational Symposium on Symbolic and Algebraic Computation (ISSAC) (2010), 25–28.

X. Shi, X. Jia, and R. Goldman. 2013. Using a bihomogeneous resultant to find the singularities of rational space curves. *J*ournal of Symbolic Computation 53 (2013), 1–25.

C. Wang. 1981. Shape classification of the parametric cubic curve and parametric B-spline cubic curve. *C*omputer-Aided Design 13, 4 (1981), 199–206.

H. Wang, X. Jia, and R. Goldman. 2009. Axial moving planes and singularities of rational space curves. *C*omputer Aided Geometric Design 26 (2009), 300–316.

X. Wang and F. Chen. 2012. Implicitization, parameterization and singularity computation of Steiner surfaces using moving surfaces. *J*ournal of Symbolic Computation 47, 6 (2012), 733–750.

A. Wong, D. Eberle, and T. Kim. 2018. Clean cloth inputs: Removing character self-intersections with volume simulation. *A*CM SIGGRAPH Poster (2018).

J. Xu, Y. Sun, and L. Zhang. 2015. A mapping-based approach to eliminating self-intersection of offset paths on mesh surfaces for CNC machining. *C*omputer-Aided Design 62 (2015), 131–142.

Z. Zeng. 2008. ApaTools: A software toolbox for approximate polynomial algebra. *A*CM Communications in Computer Algebra 42, 3 (2008), 177–179.

Q. Zhou, E. Grinspun, D. Zorin, and A. Jacobson. 2016. Mesh arrangements for solid geometry. *A*CM Transactions on Graphics 35, 4, article 39 (2016).

Received April 2021; revised June 2022; accepted July 2022