Parametrization-Invariant Marching Method for SSI

Kang Zhao
Department of Mathematics
University of Utah
Salt Lake City, UT 84112, USA
March, 1995

ABSTRACT

A marching method is presented for the computation of the intersection of two generic parametric surfaces. In the marching method, preimage curves are parametrized by the arc length of the intersection curve. Algorithms are provided for constructing predictors of high precision, adjusting the marching step size, and computing the tangent directions at double points and triple points.

Key Words: marching method, marching step size, double points, triple points, parametric surfaces, predictor, surface-surface intersection

1. Introduction

The so-called surface-surface intersection problem is to calculate the two preimages in the 2D domains of the intersection curves of two surfaces in the 3D image space. The surface-surface intersection problem has been of interest in geometric modeling for decades and a number of methods have been proposed and implemented. A comprehensive list of references on the surface-surface intersection problem can be found in An SSI Bibliography [Farin, 92]. Essentially there are two types of methods for the surface-surface intersection problem: marching method and divide-and-conquer method by the subdivision technique. In this paper we are interested in the marching method only.

There are numerous marching methods published. The essential ideas of marching methods are similar: With an initial point on an intersection curve and its two preimages in the domains some strategy is used to calculate a neighboring point on the curve and its preimages and repeat this process so that sufficiently many points and their preimages are calculated. Therefore marching method is also called curve-tracing method. Most of the strategies used in marching methods are of the same type: predictor and corrector. As it converges quadratically with a proper initial guess, the Newton-Raphson iteration is chosen as a corrector in almost all marching methods. Comparing the known marching methods, one will see various predictors are constructed according to various considerations in applications. As usual, robustness and efficiency are the two factors considered in the construction of predictors. Especially for those whose corrector is based on the Newton-Raphson iteration, the construction of an accurate predictor is critically important for the robustness. Two types of predictors have been seen in the literature. One is to predict the neighboring point on the intersection curve in the 3D space and then to obtain its preimages in both domains. For convenience, we call the predictors of this type 3D-predictor. As shown in [Faux et al, 79], the preimages might be found via calculating the intersection point of the two surfaces with another auxiliary surface through Newton-Raphson iteration. The resulting Newton-Raphson iteration involves a linear system of size 6×6(or 4×4 if the auxiliary surface is a sphere). Hence it is quite expensive to solve the resulting linear system and to adjust the marching step size for the convergence of the Newton-Raphson iteration. [Barnhill et al, 87] and [Barnhill et al, 90] proposed a method to find the preimages by projecting a point on the tangent line of the intersection curve to the two surfaces in the 3D space. Therefore the correctness of the projection will affect the robustness. The other type of predictor is to predict the preimages of the neighboring point directly in the 2D domains as did [Bajaj et al, 88] [Bajaj et al, 94] [Chen et al, 88]. We call the predictors of the latter type 2D-predictor. Our predictor is a 2D-predictor. We prefer 2D-predictors because our objective is to compute the preimages and we believe that 2D-predictors are more efficient than 3D-predictors. As did [Bajaj et al, 94], we construct predictors via the Taylor expansion formula around a point on the preimages and parametrize the preimages by the arc length of the corresponding intersection curve. Our algorithms for computing the derivatives of the preimages are similar to the one of [Bajaj et al, 94]. [Bajaj et al, 94] provided only the algorithms for computing the derivatives of order ≤ 4. As we shall see in Section 3, the derivatives of the preimages of order ≥ 5 can be computed when the curvature of the intersection curve does not vanish.

It is common to construct predictors via the Taylor expansion formula for the surface-surfaces intersection problem. It is clear that the truncation error in the Taylor expansions depends on the parametrization of the preimages. As we parametrize the preimages by the arc length of the intersection curve, the resulting predictors do not depend on the parametrizations of the surfaces. Parametrization-invariance is not the only advantage of using the arc length as the common parameter for both preimages. It also helps solve the problem of determining the marching step size. As it is well known, choosing the marching step size properly is critical for any marching method because it determines the robustness and efficiency of the method. The sensitivity of step size is clearly determined by the parametrization of the preimages. As we choose the arc length of the intersection curve as the common parameter, the step size in our marching method is measured in the arc length of the intersection curve. Therefore it reduces the sensitivity of the step size in our marching method. An algorithm for adjusting the marching step size is given in Section 4. The algorithm is constructed to guarantee the convergence of the Newton-Raphson iteration. Compared to other algorithms for adjusting step sizes, our algorithm does not need to compute matrix norms because it only needs the first order and the second order derivatives of the preimages. In order to obtain accurate prediction and to deal with double points where two intersection curves are tangent to each other, we should use those predictors that use the derivatives of the preimages of order ≥ 2. When such a predictor is used, it does not need any extra computation to adjust the marching step size. In addition, the derivatives of the preimages also enable us to deal effectively with two types of degenerate points: double points and triple points where two or three intersection curves intersect. As we know, when the surfaces do not have the same tangent plane at an intersection point, the tangent direction of the intersection curve can be obtained from the intersection line of the two tangent planes. This method fails at the double points and the triple points at which the two tangent planes often coincide. Algorithms for computing the tangent directions at double points and triple points are provided in Section 5. In the algorithms, the tangent directions are computed by finding the zeros of a quadratic form or a cubic form, dependent on at a double point or at a triple point. As the predictor can provide a good approximation of the first order derivatives of the preimages, the correct tangent direction can be chosen accordingly. For other types of degenerate points, the method proposed by [Bürger et al, 94] is implemented. In this method, if a surface patch contains a degenerate point then it is subdivided so that the marching method can be applied to those subpatches that do not contain degenerate points. In Section 6 the computation of initial points is discussed. We implement a subdivision method to find the initial points through quadtrees. Therefore, the computation of initial points is robust but expensive. For the douple points and the triple points, our algorithms enable us to find the tangent directions so that the marching process continues. The discussion of detecting recomputations is given in Section 7.

2. Prediction

Let F and G be two regular parametric surfaces defined respectively on rectangular domains D_F and

D_G:

$$F : (u,v) \in D_F \mapsto \begin{pmatrix} xf(u,v) \\ yf(u,v) \\ zf(u,v) \end{pmatrix} \quad \text{and} \quad G : (\mu,\nu) \in D_G \mapsto \begin{pmatrix} xg(\mu,\nu) \\ yg(\mu,\nu) \\ zg(\mu,\nu) \end{pmatrix}$$

and let C be one of the intersection curves of surface F and surface G: $C \subset F \cap G$. We parametrize C with its arc length s. When F and G do not have the same tangent plane along C, C is a regular curve. I.e., C has a nonzero tangent vector everywhere. If we denote by F_u and F_v the derivatives of F in the u-parameter and in the v-parameter respectively, then the unit normal vector of F can be calculated by

$$N_F := \frac{F_u \times F_v}{\|F_u \times F_v\|}.$$

Analogously, the unit normal vector of G is

$$N_G := \frac{G_\mu \times G_\nu}{\|G_\mu \times G_\nu\|}.$$

As C lies on F and G, the derivative of C, denoted by C', is perpendicular to both N_F and N_G. Therefore, C' is parallel to the cross-product of N_F with N_G:

$$N_F \times N_G.$$

By the assumption that the two surfaces do not have the same tangent plane along C, $N_F \times N_G$ does not vanish along C. Since C is parametrized by its arc length, the length of C' equals 1. It follows that

$$T := C' = \pm \frac{N_F \times N_G}{\|N_F \times N_G\|}. \tag{2.1}$$

The choice of $\pm$ depends on the marching direction.

Denote the preimages of $C(s)$ in D_F and D_G respectively by

$$f(s) := (u(s), v(s))$$

and

$$g(s) := (\mu(s), \nu(s)).$$

Then, $C(s) = F(u(s), v(s)) = G(\mu(s), \nu(s))$. In this section we show, for a given step size h, how to predict the preimages of $C(s+h)$ in D_F and in D_G, assuming that the preimages of $C(s)$ is known.

Denote by (u', v') and (μ', ν') the derivatives of f and g against s, respectively, and use similar notation for the higher order derivatives. It is natural to predict the preimages of $C(s+h)$ by

$$f(s+h) = (u(s) + u'(s)h, v(s) + v'(s)h) + O(h^2), \tag{2.2}$$

$$g(s+h) = (\mu(s) + \mu'(s)h, \nu(s) + \nu'(s)h) + O(h^2). \tag{2.3}$$

We note that h is the step size measured in the arc length of the intersection curve C. Thus we need to calculate $(u'(s), v'(s))$ and $(\mu'(s), \nu'(s))$ for the prediction. From (2.1) we obtain

$$F_u u'(s) + F_v v'(s) = \pm \frac{N_F \times N_G}{\|N_F \times N_G\|}$$

and

$$G_\mu \mu'(s) + G_\nu \nu'(s) = \pm \frac{N_F \times N_G}{\|N_F \times N_G\|}.$$

The choice of $\pm$ depends the marching direction along the curve. Thus we can calculate (u', v') because F_u and F_v are linearly independent, and similarly for (μ', ν').

Algorithm for First Order Predictor

(1) For a given initial point $P = F(u, v) = G(\mu, \nu)$, calculate F_u, F_v, G_μ, and G_ν at (u, v) and (μ, ν) respectively.
(2) Find the normal N_F and the normal N_G: $N_F = \frac{F_u \times F_v}{\|F_u \times F_v\|}$, $N_G = \frac{G_\mu \times G_\nu}{\|G_\mu \times G_\nu\|}$.
(3) Find the unit tangent vector: $T = N_F \times N_G / \|N_F \times F_G\|$.
(4) Find u' and v' by solving $T = F_u u' + F_v v'$.
(5) Find μ' and ν' by solving $T = G_\mu \mu' + G_\nu \nu'$.
(6) Update (u, v): $u = u + u'h$ and $v = v + v'h$.
(7) Update (μ, ν): $\mu = \mu + \mu' h$ and $\nu = \nu + \nu' h$.

In implementation one should check whether $\|N_F \times N_G\|$ is less than a given positive number that controls the tolerance of the smallest angle between the two tangent planes. For the other marching direction the role of surface F and surface G should be switched.

The above-presented first-order predictor can be implemented when we are only provided with the function evaluators and the first-order derivative evaluators. As the two preimages are planar curves, their geometry information cannot be predicted adequately if their second-order derivatives are not used in the prediction. Especially, when one of the preimages has a large curvature at the current point and the step size is not sufficiently small, the first-order predictor may result in divergence of the corrector which usually is constructed based on the Newton-Raphson iteration. As pointed out early, the first order predictor cannot deal with double points where two intersection curves are tangent to each other. Therefore, it is of necessity to have a second-order predictor which in addition uses the knowledge of the curvatures of the preimages when the second-order derivative evaluators are available.

When F and G have continuous second order derivatives, we can compute the second order derivatives of u, v, μ, and ν as follows. Note that

$$C''(s) = F_u u''(s) + F_v v''(s) + A_F(s)$$

and

$$C''(s) = G_\mu \mu''(s) + G_\nu \nu''(s) + A_G(s),$$

where

$$A_F(s) = F_{uu}(u'(s))^2 + 2F_{uv} u'(s) v'(s) + F_{vv}(v'(s))^2$$

and

$$A_G(s) = G_{\mu\mu}(\mu'(s))^2 + 2G_{\mu\nu}\mu'(s)\nu'(s) + G_{\nu\nu}(\nu'(s))^2$$

are known. Since $C''(s) = T'(s)$ and $T(s)$ is a unit vector, $T'(s)$ is orthogonal to $T(s)$. Therefore $T'(s)$ is on the plane spanned by the normal vectors N_F and N_G:

$$T'(s) = F_u u''(s) + F_v v''(s) + A_F(s) = c_F N_F + c_G N_G \tag{2.4}$$

and

$$T'(s) = G_\mu \mu''(s) + G_\nu \nu''(s) + A_G(s) = c_F N_F + c_G N_G \tag{2.5}$$

for some constant scalars c_F and c_G. Therefore the curvature of $C(s)$ is

$$\kappa = \|T'\| = \|c_F N_F + c_G N_G\|. \tag{2.6}$$

Multiplying (2.4) and (2.5) by N_F and N_G respectively, we can find c_F and c_G via solving

$$\begin{pmatrix} 1 & \langle N_F, N_G\rangle \\ \langle N_G, N_F\rangle & 1 \end{pmatrix} \begin{pmatrix} c_F \\ c_G \end{pmatrix} = \begin{pmatrix} \langle N_F, A_F(s)\rangle \\ \langle N_G, A_G(s)\rangle \end{pmatrix}. \tag{2.7}$$

Here and below, $\langle \cdot, \cdot\rangle$ represents the dot product. As the matrix on the left side of (2.7) is symmetric and positive definite, one can use Gaussian elimination without pivoting to solve (2.7). Thus we can obtain $(u''(s), v''(s))$ and $(\mu''(s), \nu''(s))$ by solving

$$F_u u''(s) + F_v v''(s) = c_F N_F + c_G N_G - A_F(s)$$

and

$$G_\mu \mu''(s) + G_\nu \nu''(s) = c_F N_F + c_G N_G - A_G(s).$$

Algorithm for Second Order Predictor

(1) Carry out (1) to (5) in the above Algorithm for First Order Predictor.
(2) Compute $A_F = F_{uu}(u')^2 + 2F_{uv}u'v' + F_{vv}(v')^2$.
(3) Compute $A_G = G_{\mu\mu}(\mu')^2 + 2G_{\mu\nu}\mu'\nu' + G_{\nu\nu}(\nu')^2$.
(4) Compute c_F and c_G from (2.7).
(5) Find u'' and v'' by solving $F_u u'' + F_v v'' = c_F N_F + c_G N_G - A_F$.
(6) Find μ'' and ν'' by solving $G_\mu \mu'' + G_\nu \nu'' = c_F N_F + c_G N_G - A_G$.
(7) Update (u, v): $u = u + u'h + u''h^2/2$ and $v = v + v'h + v''h^2/2$.
(8) Update (μ, ν): $\mu = \mu + \mu'h + \mu''h^2/2$ and $\nu = \nu + \nu'h + \nu''h^2/2$.

Since $\langle N_F, A_F\rangle = \langle N_F, T'\rangle$ and $\langle N_G, A_G\rangle = \langle N_G, T'\rangle$, their absolute values are the normal curvature of C on the corresponding surface. From the algorithm we see that when the first order derivatives of the preimages are known, the second order derivatives can be computed even if the two surfaces have the same tangent plane at this point.

3. Calculation of High Order Derivatives

When the construction of predictors of higher order is in order, one needs an algorithm to calculate the higher order derivatives. We in this section show how to calculate the high order derivatives of the preimages of the intersection curve C, assuming that the curvature of C does not vanish. We first illustrate how to calculate the third order derivatives of the preimages.

Calculation of the Third Order Derivatives

Let $N(s)$ and $B(s)$ be respectively the principal normal vector and the binormal vector of $C(s)$ and denote by $\kappa(s)$ and $\tau(s)$ respectively the curvature and the torsion of $C(s)$. Then by the Frenet equations,

$$\begin{pmatrix} T' \\ N' \\ B' \end{pmatrix} = \begin{pmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{pmatrix} \begin{pmatrix} T \\ N \\ B \end{pmatrix}.$$

Differentiating T' results

$$T'' = \kappa' N + \kappa(-\kappa T + \tau B) = -\kappa^2 T + \kappa' N + \kappa\tau B.$$

It follows that $T'' + \kappa^2 T$ is perpendicular to T. Thus it is a linear combination of N_F and N_G, while κ and T are known from (2.1) and (2.6). Therefore, there are two constant scalars c_F and c_G such that

$$T'' + \kappa^2 T = c_F N_F + c_G N_G.$$

On the other hand we have that

$$T'' = F_u u''' + F_v v''' + A_F(s)$$

and

$$T'' = G_\mu \mu''' + G_\nu \nu''' + A_G(s),$$

where

$$\begin{aligned} A_F(s) =& F_{uuu}(u')^3 + 3F_{uuv}(u')^2 v' + 3F_{uvv}u'(v')^2 + F_{vvv}(v')^3 + \\ & 3F_{uu}u'u'' + 3F_{uv}(u''v' + u'v'') + 3F_{vv}v'v'' \end{aligned}$$

and

$$\begin{aligned} A_G(s) =& G_{\mu\mu\mu}(\mu')^3 + 3G_{\mu\mu\nu}(\mu')^2 \nu' + 3G_{\mu\nu\nu}\mu'(\nu')^2 + G_{\nu\nu\nu}(\nu')^3 + \\ & 3G_{\mu\mu}\mu'\mu'' + 3G_{\mu\nu}(\mu''\nu' + \mu'\nu'') + 3G_{\nu\nu}\nu'\nu''. \end{aligned}$$

Note that T is perpendicular to both N_F and N_G. We can solve the following system for c_F and c_G:

$$\begin{pmatrix} 1 & \langle N_F, N_G \rangle \\ \langle N_F, N_G \rangle & 1 \end{pmatrix} \begin{pmatrix} c_F \\ c_G \end{pmatrix} = \begin{pmatrix} \langle N_F, A_F \rangle \\ \langle N_G, A_G \rangle \end{pmatrix}.$$

After finding c_F and c_G we can compute the third order derivative of μ, v, μ, and ν from

$$F_u u''' + F_v v''' = c_F N_F + c_G N_G - \kappa^2 T - A_F$$

and

$$G_\mu \mu''' + G_\nu \nu''' = c_F N_F + c_G N_G - \kappa^2 T - A_G.$$

After obtaining c_F and c_G, we comput κ' and τ as follows:

$$\kappa' = \langle c_F N_F + c_G N_G, N \rangle,$$

$$\kappa\tau = \langle c_F N_F + c_G N_G, B \rangle.$$

Hence, we need to assume that $\kappa \neq 0$ in order to compute τ.

Calculation of High Order Derivatives

The method for the calculation of derivatives of order higher than the third is similar to the one for the third order derivatives. For completeness we use the induction method on the order. From the above we see that

$$T''' = (-3\kappa\kappa')T + (\kappa'' - \kappa^3 - \kappa\tau^2)N + (\kappa\tau' + 2\kappa'\tau)B.$$

Assume that, for $j \geq 2$, we have calculated $\kappa^{(i)}$ for $i = 0, 1, ..., j-1$ and $\tau^{(i)}$ for $i = 0, 1, ..., j-2$ and that

$$T^{(j)}(s) = aT(s) + (\kappa^{(j-1)}(s) + b)N(s) + (\kappa(s)\tau^{(j-2)}(s) + c)B(s),$$

where a, b, and c are three polynomials in $\kappa^{(i)}$ for $i = 0, 1, ..., j-2$ and $\tau^{(i)}$ for $i = 0, 1, ..., j-3$. When $j = 3$, $a = -3\kappa\kappa'$, $b = -\kappa^3 - \kappa\tau^2$, and $c = 2\kappa'\tau$. Differentiating $T^{(j)}$ yields

$$\begin{aligned} T^{(j+1)} =& a'T + a\kappa N \\ &+ (\kappa^{(j)} + b')N + (\kappa^{(j-1)} + b)(-\kappa T + \tau B) \\ &+ (\kappa\tau^{(j-1)} + \kappa'\tau^{(j-2)} + c')B + (\kappa\tau^{(j-2)} + c')(-\tau)N \\ =& \alpha T + (\kappa^{(j)} + \beta)N + (\kappa\tau^{(j-1)} + \gamma)B, \end{aligned}$$

where

$$\alpha = a' - \kappa(\kappa^{(j-1)} + b)$$

is polynomial in $\kappa^{(i)}$ for $i = 0, 1, ..., j-1$ and $\tau^{(i)}$ for $i = 0, 1, ..., j-2$,

$$\beta = a\kappa + b' - \tau(\kappa\tau^{(j-2)} + c)$$

is a polynomial in $\kappa^{(i)}$ for $i = 0, 1, .., j-1$ and $\tau^{(i)}$ for $i = 0, 1, ..., j-2$, and

$$\gamma = \tau(\kappa^{(j-1)} + b) + \kappa'\tau^{(j-2)} + c'$$

is a polynomial in $\kappa^{(i)}$ for $i = 0, 1, ..., j-1$ and $\tau^{(i)}$ for $i = 0, 1, ..., j-2$. This shows that αT is known and $T^{(j+1)} - \alpha T$ is on the plane spanned by N_F and N_G. Let

$$T^{(j+1)} - \alpha T = c_F N_F + c_G N_G$$

for some constant scalars c_F and c_G. On the other hand, we have

$$T^{(j+1)} = F_u u^{(j+1)} + F_v v^{(j+1)} + A_F$$

with A_F a known vector independent of $u^{(j+1)}$ and $v^{(j+1)}$, and

$$T^{(j+1)} = G_\mu \mu^{(j+1)} + G_\nu \nu^{(j+1)} + A_G$$

with A_G a known vector independent of $\mu^{(j+1)}$ and $\nu^{(j+1)}$. From

$$\begin{pmatrix} 1 & \langle N_F, N_G\rangle \\ \langle N_F, N_G\rangle & 1 \end{pmatrix} \begin{pmatrix} c_F \\ c_G \end{pmatrix} = \begin{pmatrix} \langle N_F, A_F(s)\rangle \\ \langle N_G, A_G(s)\rangle \end{pmatrix}$$

we can find c_F and c_G. Then we can obtain $u^{(j+1)}$, $v^{(j+1)}$, $\mu^{(j+1)}$, and $\nu^{(j+1)}$ from

$$F_u u^{(j+1)} + F_v v^{(j+1)} = c_F N_F + c_G N_G + \alpha T - A_F(s)$$

and

$$G_\mu \mu^{(j+1)} + G_\nu \nu^{(j+1)} = c_F N_F + c_G N_G + \alpha T - A_G(s).$$

In addition, we can calculate $\kappa^{(j)}$ and $\tau^{(j-1)}$ from

$$(\kappa^{(j)} + \beta)N + (\kappa\tau^{(j-1)} + \gamma)B = c_F N_F + c_G N_G$$

when the curvature κ does not vanish along C:

$$\kappa^{(j)} = \langle c_F N_F + c_G N_G, N\rangle - \beta;$$

$$\kappa\tau^{(j-1)} = \langle c_F N_F + c_G N_G, B\rangle - \gamma.$$

Thus we can compute all of the derivatives of the preimages of C. We note that $u^{(j)}$ and $v^{(j)}$ can be computed if $a = \langle T^{(j)}, T\rangle$ is known and we assume that $\kappa \neq 0$ only for computing τ and its derivatives. When $j = 3$, $a = -3\kappa\kappa'$ is known. Therefore we can calculate the derivatives of the preimages of order ≤ 4 without the condition that the curvature of the intersection curve C not vanish.

4. Corrector and the Newton-Raphson Iteration

As the marching has to proceed simultaneously for the preimages in D_F and D_G, a corrector must be adopted. In the following we illustrate how to construct a corrector with respect to the predictor of second order. Let $w = (w_\mu, w_\nu)$ be a directional vector in $\mathbb{R}^2$. To correct the following values:

$$u_0 = u(s) + u'(s)h + u''(s)h^2/2,$$

$$v_0 = v(s) + v'(s)h + v''(s)h^2/2,$$

$$\mu_0 = \mu(s) + \mu'(s)h + \mu''(s)h^2/2,$$

and

$$\nu_0 = \nu(s) + \nu'(s)h + \nu''(s)h^2/2,$$

we solve the following nonlinear equation

$$F(u,v) - G(\mu_0 + w_\mu t, \nu_0 + w_\nu t) = 0 \tag{4.1}$$

for (u, v, t) with initial guess: $(u_0, v_0, 0)$. It is clear that

$$\mu = \mu_0 + w_\mu t \quad \text{and} \quad \nu = \nu_0 + w_\nu t$$

describe the line passing through (μ_0, ν_0) and perpendicular to

$$w^\perp := (w_\nu, -w_\mu).$$

A geometric interpretation of (4.1) is to find the intersection of the surface F with the curve on the surface G described by $\mu = \mu_0 + w_\mu t$ and $\nu = \nu_0 + w_\nu t$ in a neighborhood of $C(s)$. The corresponding Newton-Raphson iteration scheme is:

$$\begin{pmatrix} u_{n+1} \\ v_{n+1} \\ t_{n+1} \end{pmatrix} = \begin{pmatrix} u_n \\ v_n \\ t_n \end{pmatrix} - \begin{pmatrix} \Delta u_{n+1} \\ \Delta v_{n+1} \\ \Delta t_{n+1} \end{pmatrix} \tag{4.2}$$

with

$$F_u(u_n, v_n)\Delta u_{n+1} + F_v(u_n, v_n)\Delta v_{n+1} - (G_\mu(\mu_n, \nu_n)w_\mu + G_\nu(\mu_n, \nu_n)w_\nu)\Delta t_{n+1} = F(u_n, v_n) - G(\mu_n, \nu_n).$$

Correspondingly, (μ, ν) is updated by

$$\mu_{n+1} = \mu_n + w_\mu t_{n+1}, \quad \nu_{n+1} = \nu_n + w_\nu t_{n+1}.$$

The directional vector w should be chosen properly so that the Jacobian is nonsingular. We choose the directional vector w as follows:

$$w_\mu = -\nu'(s) \quad \text{and} \quad w_\nu = \mu'(s).$$

Namely, w is a vector perpendicular to $g'(s) = (\mu'(s), \nu'(s))$ and parallel to the normal vector of g at s. As the tangent vector of the intersection curve at s

$$G_\mu \mu'(s) + G_\nu \nu'(s)$$

lies on the tangent plane spanned by F_u and F_v, if we assume that the two tangent planes do not coincide at this point then

$$-G_\mu \nu'(s) + G_\nu \mu'(s)$$

will not lie on the tangent plane spanned by F_u and F_v at this point because the matrix

$$\begin{pmatrix} \mu'(s) & \nu'(s) \\ -\nu'(s) & \mu'(s) \end{pmatrix}$$

is nonsingular. This proves that the Jacobian for (4.2) is not singular at least for the first step of iteration when the two tangent planes do not coincide at this point.

As the predictor is of second order, our experience shows that two steps of the Newton-Raphson iteration are usually sufficient for $h \leq 0.1$: One step of the Newton-Raphson iteration with the above scheme and another step of the Newton-Raphson iteration in which the role of surface F and surface G are switched.

Algorithm for Corrector

(1) Compute $(\Delta u, \Delta v, \Delta t)$ from solving

$$\begin{aligned} F_u(u_0, v_0)\Delta u + F_v(u_0, v_0)\Delta v - (-G_\mu(\mu_0, \nu_0)\nu' + G_\nu(\mu_0, \nu_0)\mu')\Delta t \\ = F(u_0, v_0) - G(\mu_0, \nu_0). \end{aligned}$$

(2) Update (u, v) and (μ, ν):

$$\begin{pmatrix} u \\ v \\ \mu \\ \nu \end{pmatrix} = \begin{pmatrix} u_0 - \Delta u \\ v_0 - \Delta v \\ \mu_0 + \Delta t\ \nu' \\ \nu_0 - \Delta t\ \mu' \end{pmatrix}. \tag{4.3}$$

(3) If $\|F(u, v) - G(\mu, \nu)\|$ is less than or equal to the error tolerance then stop.

(4) Compute $(\Delta \mu, \Delta \nu, \Delta t)$ from solving

$$G_\mu(\mu, \nu)\Delta\mu + G_\nu(\mu, \nu)\Delta\nu - (-F_u(u, v)v' + F_v(u, v)u')\Delta t = G(\mu, \nu) - F(u, v).$$

(2) Update (u, v) and (μ, ν):

$$\begin{pmatrix} \mu \\ \nu \\ u \\ v \end{pmatrix} = \begin{pmatrix} \mu - \Delta\mu \\ \nu - \Delta\nu \\ u + \Delta t\ v' \\ v - \Delta t\ u' \end{pmatrix}. \tag{4.4}$$

How to Adjust the Step Size

The step size is known to be critical for any marching method because it determines the robustness and the efficiency. As the Newton-Raphson iteration is used as a corrector and the Newton-Raphson iteration has local convergence only, the step size must be adjusted at each step to guarantee the convergence and to make sure the marching process advances along each intersection curve completely. [Bürger et al, 93] has considered the choice of step size for the convergence of Newton-Raphson iteration. The step size also determines the number and the positions of the points to be computed on the intersection curve. As a marching method calculates a finite number of points on the intersection curve only, eventually one needs some interpolation scheme through these points to approximate the intersection curve. Thus the step size also determines the approximation to the intersection curve. One known method for choosing the step size is to choose an angle with respect to the osculating circles for the intersection curve and the two preimages [Faux et al, 79]. Thus the computation or an accurate approximation of the second order derivatives of the preimages is necessary for choosing the step size. As the step size in our method is measured in the arc length of the intersection curve, we do not need to appeal to the osculating circles, even for the case where the first order predictor is used.

The new feature of our algorithm for adjusting the marching step size is that it concerns only the two preimages of the intersection curve in the parameter domains and does not need to compute matrix norms. The expense of the technique is to compute the second order derivatives and it seems inevitable for adjusting the step size. As pointed out earlier, it is necessary to calculate the second order derivatives of the preimages in order to predict the geometry of the intersection curve and its preimages.

As the Newton-Raphson iteration is used to find the intersection point of the surface F with the curve on surface G corresponding to

$$\mu = \mu_0 - t\nu'(s), \quad \nu = \nu_0 + t\mu'(s), \tag{4.5}$$

with

$$u_0 = u(s) + u'(s)h + u''(s)h^2/2,$$

$$v_0 = v(s) + v'(s)h + v''(s)h^2/2,$$

$$\mu_0 = \mu(s) + \mu'(s)h + \mu''(s)h^2/2,$$

$$\nu_0 = \nu(s) + \nu'(s)h + \nu''(s)h^2/2.$$

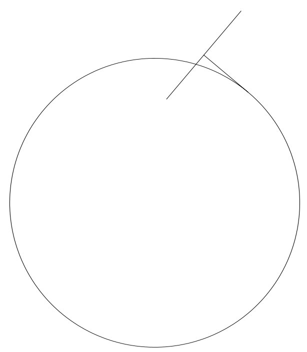

Fig. 1.

The parameters (μ^*, ν^*) of the intersection point on C in the domain D_G is the intersection point of the preimage in D_G with the straight line (4.5), see Fig. 1. If we have the formulas of the $\mu(s)$ and the $\nu(s)$, then we can use the Newton-Raphson iteration to find the intersection point (μ^*, ν^*) first then determine the parameters in D_F. To find (μ^*, ν^*) we need to solve

$$\begin{cases} \mu(s+\Delta s) - \mu_0 + \Delta t \nu'(s) = 0 \\ \nu(s+\Delta s) - \nu_0 - \Delta t \mu'(s) = 0 \end{cases} \tag{4.6}$$

for $(\Delta s, \Delta t)$ with initial guess $(0,0)$. The Newton-Raphson iteration scheme gives

$$\begin{pmatrix} \Delta s \\ \Delta t \end{pmatrix} = -J^{-1} \begin{pmatrix} \mu(s) - \mu_0 \\ \nu(s) - \nu_0 \end{pmatrix},$$

where J is the Jacobian:

$$J = \begin{pmatrix} \mu'(s) & \nu'(s) \\ \nu'(s) & -\mu'(s) \end{pmatrix}.$$

When $\mu'(s) \neq 0$ or $\nu'(s) \neq 0$, J is invertible and

$$J^{-1} = \frac{1}{(\mu'(s))^2 + (\nu'(s))^2} \begin{pmatrix} \mu'(s) & \nu'(s) \\ \nu'(s) & -\mu'(s) \end{pmatrix}. \tag{4.7}$$

As $\mu_0 - \mu(s) = \mu'(s)h + \mu''(s)h^2/2$ and $\nu_0 - \nu(s) = \nu'(s)h + \nu''(s)h^2/2$, we therefore obtain that

$$\begin{pmatrix} \Delta s \\ \Delta t \end{pmatrix} = \frac{1}{(\mu'(s))^2 + (\nu'(s))^2} \begin{pmatrix} \mu'(s) & \nu'(s) \\ \nu'(s) & -\mu'(s) \end{pmatrix} \begin{pmatrix} \mu'(s)h + \mu''(s)h^2/2 \\ \nu'(s)h + \nu''(s)h^2/2 \end{pmatrix}.$$

Therefore,

$$\Delta s = h + \frac{\mu'(s)\mu''(s) + \nu'(s)\nu''(s)}{(\mu'(s))^2 + (\nu'(s))^2} h^2/2 \tag{4.8}$$

and

$$\Delta t = \frac{\nu'(s)\mu''(s) - \mu'(s)\nu''(s)}{(\mu'(s))^2 + (\nu'(s))^2} h^2/2. \tag{4.9}$$

Define

$$\varphi(\Delta s, \Delta t) := \mu(s+\Delta s) - \mu_0 + \Delta t \nu'(s)$$

and

$$\psi(\Delta s, \Delta t) := \nu(s+\Delta s) - \nu_0 - \Delta t \mu'(s).$$

Assume that $(\Delta s^*, \Delta t^*)$ is the solution for (4.6). Then $\varphi(\Delta s^*, \Delta t^*) = 0$ and $\psi(\Delta s^*, \Delta t^*) = 0$, and

$$E_1 := \begin{pmatrix} \Delta s^* - \Delta s \\ \Delta t^* - \Delta t \end{pmatrix} = \begin{pmatrix} \Delta s^* \\ \Delta t^* \end{pmatrix} + J^{-1} \begin{pmatrix} \varphi(0,0) - \varphi(\Delta s^*, \Delta t^*) \\ \psi(0,0) - \psi(\Delta s^*, \Delta t^*) \end{pmatrix}.$$

Let

$$H_\varphi := \begin{pmatrix} \mu''(s) & \mu'(s) + \nu'(s) \\ \mu'(s) + \nu'(s) & 0 \end{pmatrix}$$

and

$$H_\psi := \begin{pmatrix} \nu''(s) & \nu'(s) - \mu'(s) \\ \nu'(s) - \mu')(s) & 0 \end{pmatrix}.$$

Denote $E_0 := (\Delta s^*, \Delta t^*)^T$. By the Taylor expansions of φ and ψ,

$$E_1 = \frac{1}{2} J^{-1} \begin{pmatrix} E_0^T H_\varphi E_0 \\ E_0^T H_\psi E_0 \end{pmatrix} + O(h^3).$$

It follows that

$$\begin{aligned}\|E_1\|_2 &\le \frac{1}{2}\|J^{-1}\|_2\sqrt{|E_0^T H_\varphi E_0|^2 + |E_0^T H_\psi E_0|^2} + O(h^3)\\ &\le \frac{1}{2}\|J^{-1}\|_2\|E_0\|_2^2\sqrt{\|H_\varphi\|_2^2 + \|H_\psi\|_2^2} + O(h^3).\end{aligned}$$

Discarding the term $O(h^3)$, we can expect the convergence of the Newton-Raphson iteration when

$$\frac{1}{2}\|J^{-1}\|_2\|E_0\|_2\sqrt{\|H_\varphi\|_2^2 + \|H_\psi\|_2^2} < 1. \tag{4.10}$$

For small h, $(\Delta s, \Delta t)^T$ should be close to $E_0 = (\Delta s^*, \Delta t^*)^T$. For practical applications, we may replace E_0 by the vector $(\Delta s, \Delta t)^T$ for the purpose of adjusting the step size h. From (4.8) and (4.9) we know that

$$\begin{aligned}\|E_0\|_2^2 &\simeq h^2 + \frac{\mu'(s)\mu''(s) + \nu'(s)\nu''(s)}{(\mu'(s))^2 + (\nu'(s))^2}h^3 + \frac{(\mu''(s))^2 + (\nu''(s))^2}{(\mu'(s))^2 + (\nu'(s))^2}h^4/4\\ &\le h^2 + \sqrt{\frac{(\mu''(s))^2 + (\nu''(s))^2}{(\mu'(s))^2 + (\nu'(s))^2}}h^3 + \frac{(\mu''(s))^2 + (\nu''(s))^2}{(\mu'(s))^2 + (\nu'(s))^2}h^4/4\\ &= h^2\left(1 + \sqrt{\frac{(\mu''(s))^2 + (\nu''(s))^2}{(\mu'(s))^2 + (\nu'(s))^2}}h/2\right)^2.\end{aligned}$$

From (4.7) we obtain

$$\|J^{-1}\|_2^2 = \frac{1}{(\mu'(s))^2 + (\nu'(s))^2}.$$

Since

$$\|H_\varphi\|_2 = \frac{|\mu''(s)| + \sqrt{|\mu''(s)|^2 + 4|\mu'(s) + \nu'(s)|^2}}{2}$$

and

$$\|H_\psi\|_2 = \frac{|\nu''(s)| + \sqrt{|\nu''(s)|^2 + 4|\mu'(s) - \nu'(s)|^2}}{2},$$

$$\begin{aligned}\|H_\varphi\|_2^2 + \|H_\psi\|_2^2 =&1/2(|\mu''(s)|^2 + |\nu''(s)|^2) + 2(|\mu'(s)|^2 + |\nu'(s)|^2)\\ &+ 1/2(|\mu''(s)|\sqrt{|\mu''(s)|^2 + 4|\mu'(s) + \nu'(s)|^2})\\ &+ 1/2(|\nu''(s)|\sqrt{|\nu''(s)|^2 + 4|\mu'(s) - \nu'(s)|^2})\\ \le&1/2(|\mu''(s)|^2 + |\nu''(s)|^2) + 2(|\mu'(s)|^2 + |\nu'(s)|^2)\\ &+ 1/4(|\mu''(s)|^2 + |\mu''(s)|^2 + 4|\mu'(s) + \nu'(s)|^2)\\ &+ 1/4(|\nu''(s)|^2 + |\nu''(s)|^2 + 4|\mu'(s) - \nu'(s)|^2)\\ =&|\mu''(s)|^2 + |\nu''(s)|^2 + 4(|\mu'(s)|^2 + |\nu'(s)|^2).\end{aligned}$$

Thus when

$$\left(4 + \frac{|\mu''(s)|^2 + |\nu''(s)|^2}{(\mu'(s))^2 + (\nu'(s))^2}\right)h^2\left(1 + \sqrt{\frac{(\mu''(s))^2 + (\nu''(s))^2}{(\mu'(s))^2 + (\nu'(s))^2}}h/2\right)^2 < 4,$$

then (4.10) holds. A simplified condition for adjusting the step size h is

$$h \le \sqrt{7}/6 \quad \text{and} \quad \frac{|\mu''(s)|^2 + |\nu''(s)|^2}{|\mu'(s)|^2 + |\nu'(s)|^2}h^2 < 1 \tag{4.11}$$

Analogously we have

$$h \le \sqrt{7}/6 \quad \text{and} \quad \frac{|u''(s)|^2 + |v''(s)|^2}{|u'(s)|^2 + |v'(s)|^2}h^2 < 1 \tag{4.12}$$

Algorithm for Adjusting Step Size h

(0) Input maxStepSize.
(1) Let $h = \min\{\sqrt{7}/6, \text{ maxStepSize }\}$.
(2) Let $\Delta_1 = (|u''(s)|^2 + |v''(s)|^2)/(|u'(s)|^2 + |v'(s)|^2)$.
(3) If $h^2\Delta_1 > 1$ then let $h = 1/\sqrt{\Delta_1}$.
(4) Let $\Delta_2 = (|\mu''(s)|^2 + |\nu''(s)|^2)/(|\mu'(s)|^2 + |\nu'(s)|^2)$.
(5) If $h^2\Delta_2 > 1$ then let $h = 1/\sqrt{\Delta_2}$.

5. Degenerate Points

By degenerate points we mean the points on the two surfaces where the two surfaces intersect with each other and have the same tangent plane. Since we in general compute the tangent vector of the intersection curve from the direction parallel to the intersection line of the two tangent planes, we must resort to another method for computing the tangent vector at degenerate points. There are a number of techniques introduced in the literature to deal with various types of degenerate points. It is well known that the subdivision method for the calculation of surface-surface intersection does not need a special treatment for degenerate points. A method proposed by [Bürger et al, 93] is implemented that subdivides the pair of patches and apply a marching method to the subpatches that do not contain degenerate points. In the implementation a stack is used to hold the pairs of the subpatches of which the diagonal of each rectangular domain is larger than a given positive number. Therefore, if the intersection curve consists of degenerate points then the intersection is actually calculated via a subdivision method. Usually, degenerate points are isolated in surface-surfaces intersection problem. Other two methods are also implemented for handling two types of isolated degenerate points: double points and triple points.

Double Points

Assume that the two surfaces have the same tangent plane at an intersection point $F(u(s), v(s)) = G(\mu(s), \nu(s))$. Then

$$F_u u' + F_v v' = G_\mu \mu' + G_\nu \nu'.$$

Thus we obtain that

$$\begin{pmatrix} \mu' \\ \nu' \end{pmatrix} = H \begin{pmatrix} u' \\ v' \end{pmatrix}, \tag{5.1}$$

where

$$H := \begin{pmatrix} \langle G_\mu, G_\mu \rangle & \langle G_\mu, G_\nu \rangle \\ \langle G_\mu, G_\nu \rangle & \langle G_\nu, G_\nu \rangle \end{pmatrix}^{-1} \begin{pmatrix} \langle G_\mu, F_u \rangle & \langle G_\mu, F_v \rangle \\ \langle G_\nu, F_u \rangle & \langle G_\nu, F_v \rangle \end{pmatrix}.$$

When F and G have continuous second order derivatives,

$$F_{uu}(u')^2 + 2F_{uv}u'v' + F_{vv}(v')^2 + T_F = G_{\mu\mu}(\mu')^2 + 2G_{\mu\nu}\mu'\nu' + G_{\nu\nu}(\nu')^2 + T_G, \tag{5.2}$$

where $T_F = F_u u'' + F_v v''$ and $T_G = G_\mu \mu'' + G_\nu \nu''$. As F and G have the same tangent plane at this point, their normal lines are parallel. Let N be a unit vector parallel to $F_u \times F_v$. Then $\langle N, T_F \rangle = \langle N, T_G \rangle = 0$. Define

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} := \begin{pmatrix} \langle N, F_{uu} \rangle & \langle N, F_{uv} \rangle \\ \langle N, F_{uv} \rangle & \langle N, F_{vv} \rangle \end{pmatrix}$$

and

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} := \begin{pmatrix} \langle N, G_{\mu\mu} \rangle & \langle N, G_{\mu\nu} \rangle \\ \langle N, G_{\mu\nu} \rangle & \langle N, G_{\nu\nu} \rangle \end{pmatrix}.$$

From (5.2) we obtain that

$$(u'v') \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix} = (\mu'\nu') \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} \mu' \\ \nu' \end{pmatrix}.$$

From (5.1) it follows that

$$c_{11}(u')^2 + 2c_{12}u'v' + c_{22}(v')^2 = (u'v') C \begin{pmatrix} u' \\ v' \end{pmatrix} = 0 \tag{5.3}$$

with

$$C := \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} - H^T \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} H. \tag{5.4}$$

Namely, (u', v') is an intersection point of the two osculating paraboloids. When $C \neq 0$ and the parametrization of $(u(s), v(s))$ is regular, the quadratic form in (5.3) is not elliptic because it has nontrivial solutions. When the quadratic form (5.3) is hyperbolic, we can find two solutions for $(u'v')$. When $c_{11} \neq 0$, the solutions are

$$u' = -c_{12} \pm \sqrt{c_{12}^2 - c_{11}c_{22}} \quad \text{and} \quad v' = c_{11}.$$

When $c_{22} \neq 0$, the solutions are

$$u' = c_{22} \quad \text{and} \quad v' = -c_{12} \pm \sqrt{c_{12}^2 - c_{11}c_{22}}.$$

In the case $c_{11} = c_{22} = 0$ and $c_{12} \neq 0$, the solutions are

$$u' = 1, \quad \text{and} \quad v' = 0$$

and

$$u' = 0, \quad \text{and} \quad v' = 1$$

Each solution (u', v') gives a tangent direction: $F_u u' + F_v v'$. When the quadratic form is parabolic, we can find one solution for (u', v'). If this is a double point, then the two intersection curves share a tangent line and any predictor that uses only the first order derivatives fails at this point. The above method for computing the tangent directions is essentially the same as [Piegl, 92] in which Piegl described it by means of the Dupin indicatrix. When two tangent directions are obtained, one can use the predictor to obtain an approximation of the derivatives of the preimages and then chooses the right tangent direction. The accuracy of the approximation depends on the predictor and the marching step size. If the second order predictor is used, the error of the approximation is of order $O(h^2)$ with h the marching step size.

Algorithm for Computing Tangent Vectors at Double Points

(0) Input the first order and the second order derivatives of the preimages at the previous point: u_0', v_0', u_0'', and v_0''.
(1) Compute the approximation: $u' = u_0' + hu_0''$ and $v' = v_0' + hv_0''$.
(2) Compute the coefficients c_{ij} of the quadratic form (5.3).
(3) Compute $\Delta = \sqrt{c_{12}^2 - c_{11}c_{22}}$.
(4) If $c_{11} \neq 0$, let $u_1' = -c_{12} + \Delta$, $v_1' = c_{11}$, $u_2' = -c_{12} - \Delta$, and $v_2' = c_{11}$. Go to (8).
(5) If $c_{22} \neq 0$, let $u_1' = c_{22}$, $v_1' = -c_{12} + \Delta$, $u_2' = c_{22}$, $v_2' = -c_{12} - \Delta$. Go to (8).
(6) If $c_{12} = 0$, return and report that this method failed.

(7) Let $u_1' = 1$, $v_1' = 0$, $u_2' = 0$, and $v_2' = 1$.

(8) If $|u'v_1' - v'u_1'| \leq |u'v_2' - v'u_2'|$ output the tangent direction $F_u u_1' + F_v v_1'$. Otherwise output the tangent direction $F_u u_2' + F_v v_2'$.

Triple Points

When the matrix $C = 0$, the two surfaces share the same osculating paraboloid. In order to compute the tangent direction vectors, we look into the third order derivatives:

$$\begin{aligned}
C''' =& F_u u''' + F_v''' \\
&+ F_{uuu}(u')^3 + 3F_{uuv}(u')^2 v' + 3F_{uvv}u'(v')^2 + F_{vvv}(v')^2 \\
&+ 3F_{uu}u'u'' + 3F_{uv}(u'v'' + u''v') + 3F_{vv}v'v'' \\
=& G_\mu \mu''' + G_\nu \nu''' \\
&+ G_{\mu\mu\mu}(\mu')^3 + 3G_{\mu\mu\nu}(\mu')^2\nu + 3G_{\mu\nu\nu}\mu'(\nu')^2 + G_{\nu\nu\nu}(\nu')^3 \\
&+ 3G_{\mu\mu}\mu'\mu'' + 3G_{\mu\nu}(\mu'\nu'' + \mu''\nu') + G_{\nu\nu}\nu'\nu''.
\end{aligned}$$

Define

$$\begin{pmatrix} a_{111} & a_{112} \\ a_{122} & a_{222} \end{pmatrix} := \begin{pmatrix} \langle N, F_{uuu}\rangle & \langle N, F_{uuv}\rangle \\ \langle N, F_{uvv}\rangle & \langle N, F_{vvv}\rangle \end{pmatrix}$$

and

$$\begin{pmatrix} b_{111} & b_{112} \\ b_{122} & b_{222} \end{pmatrix} := \begin{pmatrix} \langle N, G_{\mu\mu\mu}\rangle & \langle N, G_{\mu\mu\nu}\rangle \\ \langle N, G_{\mu\nu\nu}\rangle & \langle N, G_{\nu\nu\nu}\rangle . \end{pmatrix}$$

Let

$$p_1 := (\, u'v' \,) \begin{pmatrix} a_{111} & a_{112} \\ a_{112} & a_{122} \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix},$$

$$p_2 := (\, u'v' \,) \begin{pmatrix} a_{112} & a_{122} \\ a_{122} & a_{222} \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix},$$

$$q_1 := (\, \mu'\nu' \,) \begin{pmatrix} b_{111} & b_{112} \\ b_{112} & b_{122} \end{pmatrix} \begin{pmatrix} \mu' \\ \nu' \end{pmatrix} = (\, u'v' \,) H^T \begin{pmatrix} b_{111} & b_{112} \\ b_{112} & b_{122} \end{pmatrix} H \begin{pmatrix} u' \\ v' \end{pmatrix},$$

and

$$q_2 := (\, \mu'\nu' \,) \begin{pmatrix} b_{112} & b_{122} \\ b_{122} & b_{222} \end{pmatrix} \begin{pmatrix} \mu' \\ \nu' \end{pmatrix} = (\, u'v' \,) H^T \begin{pmatrix} b_{112} & b_{122} \\ b_{122} & b_{222} \end{pmatrix} H \begin{pmatrix} u' \\ v' \end{pmatrix}.$$

Then we have that

$$(\, u'v' \,) \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} + 3\,(\, u'v' \,) \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \begin{pmatrix} u'' \\ v'' \end{pmatrix} = (\, \mu'\nu' \,) \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} + 3\,(\, \mu'\nu' \,) \begin{pmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{pmatrix} \begin{pmatrix} \mu'' \\ \nu'' \end{pmatrix}.$$

From (5.2) it follows that

$$\begin{pmatrix} \mu'' \\ \nu'' \end{pmatrix} = H \begin{pmatrix} u'' \\ v'' \end{pmatrix} + J \begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix},$$

where

$$J := \begin{pmatrix} \langle G_\mu, G_\mu\rangle & \langle G_\mu, G_\nu\rangle \\ \langle G_\mu, G_\nu\rangle & \langle G_\nu, G_\nu\rangle \end{pmatrix}^{-1}$$

and

$$\varphi_1 := (\, u'v' \,) \left(\begin{pmatrix} \langle G_\mu, F_{uu}\rangle & \langle G_\mu, F_{uv}\rangle \\ \langle G_\mu, F_{uv}\rangle & \langle G_\mu, F_{vv}\rangle \end{pmatrix} - H^T \begin{pmatrix} \langle G_\mu, G_{\mu\mu}\rangle & \langle G_\mu, G_{\mu\nu}\rangle \\ \langle G_\mu, G_{\mu\nu}\rangle & \langle G_\mu, G_{\nu\nu}\rangle \end{pmatrix} H \right) \begin{pmatrix} u' \\ v' \end{pmatrix},$$

$$\varphi_2 := (\, u'v' \,) \left(\begin{pmatrix} \langle G_\nu, F_{uu}\rangle & \langle G_\nu, F_{uv}\rangle \\ \langle G_\nu, F_{uv}\rangle & \langle G_\nu, F_{vv}\rangle \end{pmatrix} - H^T \begin{pmatrix} \langle G_\nu, G_{\mu\mu}\rangle & \langle G_\nu, G_{\mu\nu}\rangle \\ \langle G_\nu, G_{\mu\nu}\rangle & \langle G_\nu, G_{\nu\nu}\rangle \end{pmatrix} H \right) \begin{pmatrix} u' \\ v' \end{pmatrix}.$$

As the two surfaces share the same osculating paraboloid, the matrix C defined by (5.4) is 0. Therefore,

$$(\, u'v' \,) \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \begin{pmatrix} u'' \\ v'' \end{pmatrix} = (\, u'v' \,)\, H^T \begin{pmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{pmatrix} H \begin{pmatrix} u'' \\ v'' \end{pmatrix}.$$

Thus we obtain the following equation for computing (u', v'):

$$(\, u'v' \,) \left(\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} - H^T \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} \right) = 3\,(\, u'v' \,)\, H^T \begin{pmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{pmatrix} J \begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix}.$$

Since p_1, p_2, q_1, q_2, φ_1, and φ_2 are quadratic forms of u' and v', this equation is a cubic form. Thus it may provide 3 solutions for (u', v') and, correspondingly, 3 tangent direction vectors.

Solving the Cubic Form

Denote

$$H =: \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} := H^T \begin{pmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{pmatrix} J.$$

Then,

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} - H^T \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} - 3D \begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} p_1 - h_{11}q_1 - h_{21}q_2 - 3d_{11}\varphi_1 - 3d_{12}\varphi_2 \\ p_2 - h_{12}q_1 - h_{22}q_2 - 3d_{21}\varphi_1 - 3d_{22}\varphi_2 \end{pmatrix}$$

For $i = 1, 2$, $p_i - h_{1i}q_1 - h_{2i}q_2 - 3d_{i1}\varphi_1 - 3d_{i2}\varphi_2$ is a quadratic form. Assume that

$$p_i - h_{1i}q_1 - h_{2i}q_2 - 3d_{i1}\varphi_1 - 3d_{i2}\varphi_2 = (\, u'v' \,) \begin{pmatrix} c_{11i} & c_{1i2} \\ c_{1i2} & c_{i22} \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix},$$

then

$$\begin{aligned} \begin{pmatrix} c_{11i} & c_{1i2} \\ c_{1i2} & c_{i22} \end{pmatrix} = & \begin{pmatrix} a_{11i} & a_{1i2} \\ a_{1i2} & a_{i22} \end{pmatrix} - H^T \left(h_{1i} \begin{pmatrix} b_{111} & b_{112} \\ b_{112} & b_{122} \end{pmatrix} + h_{2i} \begin{pmatrix} b_{112} & b_{122} \\ b_{122} & b_{222} \end{pmatrix} \right) H \\ & - 3 \begin{pmatrix} \langle d_{i1}G_\mu + d_{i2}G_\nu, F_{uu} \rangle & \langle d_{i1}G_\mu + d_{i2}G_\nu, F_{uv} \rangle \\ \langle d_{i1}G_\mu + d_{i2}G_\nu, F_{uv} \rangle & \langle d_{i1}G_\mu + d_{i2}G_\nu, F_{vv} \rangle \end{pmatrix} \\ & + 3H^T \begin{pmatrix} \langle d_{i1}G_\mu + d_{i2}G_\nu, G_{uu} \rangle & \langle d_{i1}G_\mu + d_{i2}G_\nu, G_{uv} \rangle \\ \langle d_{i1}G_\mu + d_{i2}G_\nu, G_{uv} \rangle & \langle d_{i1}G_\mu + d_{i2}G_\nu, G_{vv} \rangle \end{pmatrix} H. \end{aligned}$$

Thus the cubic form is

$$c_{111}(u')^3 + 3c_{112}(u')^2 v' + 3c_{122}u'(v')^2 + c_{222}(v')^3 = 0. \tag{5.5}$$

If there are three intersection curves intersect at this point, the cubic form (5.5) determines three tangent directions. As we can obtain approximation u'_0 and v'_0 of u' and v', we can use the cubic form to refine u'_0 and v'_0 via the Newton-Raphson iteration, instead of finding the three zeros of the cubic form by an analytic formula. For instance, when $|v'_0| \geq |u'_0|$, let $w_0 = u'_0/v'_0$. Then we solve

$$c_{111}w^3 + 3c_{112}w^2 + 3c_{122}w + c_{222} = 0 \tag{5.6}$$

with the initial guess w_0. The corresponding Newton-Raphson iteration scheme is, for $n = 0, 1, ..., k-1$,

$$w_{n+1} = w_n - \frac{c_{111}w_n^3 + 3c_{112}w_n^2 + 3c_{122}w_n + c_{222}}{3(c_{111}w_n^2 + 2c_{112}w_n + c_{122}w_n)}.$$

The corresponding tangent direction is $F_u w_k + F_u$.

Algorithm for Computing Tangent Vectors at Triple Points

(0) Input the first order and the second order derivatives of the preimages at the previous point: u_0', v_0', u_0'', and v_0''.

(1) Compute the approximation: $u' = u_0' + hu_0''$ and $v' = v_0' + hv_0''$.

(2) Compute the coefficients c_{ij} of the cubic form (5.6). If all coefficients $c_{ij} = 0$ then return and report that this method failed.

(3) If $|v'| < |u'|$ go to (6).

(4) Let $w_0 = u'/v'$ and do the following Newton-Raphson iterations:

$$w_{n+1} = w_n - \frac{c_{111}w_n^3 + 3c_{112}w_n^2 + 3c_{122}w_n + c_{222}}{3(c_{111}w_n^2 + 2c_{112}w_n + c_{122}w_n)}$$

for $n = 0, 1, ..., k-1$.

(5) Output tangent direction $F_u w_k + F_v$ and stop.

(6) Let $w_0 = v'/u'$ and do the following Newton-Raphson iterations:

$$w_{n+1} = w_n - \frac{c_{222}w_n^3 + 3c_{122}w_n^2 + 3c_{112}w_n + c_{111}}{3(c_{222}w_n^2 + 2c_{122}w_n + c_{112}w_n)}$$

for $n = 0, 1, ..., k-1$.

(7) Output tangent direction $F_u + F_v w_k$ and stop.

6. Calculation of Initial Points

Any marching method for the calculation of surface-surface intersection needs an auxiliary method to find an initial point on each intersection curve. In our implementation we use the subdivision method to find the initial points. The technique of subdivision itself can constitute a method for the computation of surface-surface intersection for thoses surfaces that have the convex hull property such as tensor-product B-spline surfaces, because the subdivision method depends of the availability of the bounding boxes [Houghton et al, 85].

In the calculation of initial points, bounding boxes were used in numerous marching methods to exclude nonintersecting subpatches, in attempt to promote efficiency. In the case of generic parametric surfaces, we in general are not able to compute the bounding boxes. Therefore, it is very possible to exclude subpatchs that actually have a nonempty intersection because the bounding boxes do not completely bound the surface subpatches. For this reason we do not use bounding boxes to test whether two subpatches are disjoint. We only use very tight bounding boxes to test whether two subpatches are joint. As usual, each rectangular domain is subdivided into identical rectangles. For each small rectangle we use the image of its four vertices to form a parallelepiped as a bounding box. Perhaps we should not use the terminology of bounding boxes because they are not supposed to bound the corresponding subpatches at all. The boxes are used just because it is relatively easier to examine whether two 3D parallelepipeds intersect. However we still use this terminology in this paper.

As our purpose of using subdivision is solely for the calculation of initial points, we implement it as does a bisection method. As it is well known, the bisection method for finding a zero of a univariate continuous function is very robust when the function has a unique zero and changes its sign in an interval. Recall that in the bisection method one subdivides the current interval into two intervals at the mid-point of the interval and applies the bisection method to one of the two intervals on which the function changes its sign. When the function has a unique zero in the interval, then one only needs to check the sign of the function values at the end points of the interval. While the function has more than one zeros in the interval, it may not be sufficient to check the function values at the end points only. The calculation of initial points on an intersection curve of the surfaces is sort of similar to finding a zero of a univariate function that has more than one zeros on a given interval via the bisection method. At first each rectangular domain is subdivided into four identical rectangles and then the bounding boxes are used to choose one rectangle from each domain on which the corresponding surface patch potentially has a nonempty intersection with another surface patch. Because of the uncertainty of the disjointness of the subpatches on the subrectangular domains, we cannot abandon any subpatches whose bounding boxes have an empty intersection. Therefore, we implement our subdivision via quadtrees, as did [Barnhill et al, 90]. The nodes in the quadtrees represent the subpatches and the number of the level of the quadtrees indicates the accuracy of the approximation of the initial points. For a given level control, if at a leaf node on the final level the bounding box intersects with the bounding box of a leaf node in the other quadtree then an initial point has been found. Otherwise, it will be reported that the two surfaces have an empty intersection.

7. Detection of Recomputations

As pointed out by a number of authors, in order to make a marching method efficient one needs an effective algorithm to detect recomputations of the intersection curves. As there may exist branch points on the surfaces where two or more intersection curves join together, comparison of the position coordinates is apparently not sufficient. Therefore, the use of certain orders of derivatives can help us to detect the recomputation.

We save the discrete position coordinates of the intersection curves and their preimages together with their derivatives in linked lists. Each node in the linked lists contains:

the position coordinates (x_F, y_F, z_F) on the intersection curve and on the surface F;
the parameter values (u, v) of the preimage of (x_F, y_F, z_F) in the domain D_F;
the first order derivative (u', v');
the position coordinates (x_G, y_G, z_G) on the intersection curve and on the surface G;
the parameter values (μ, ν) of the preimage of (x_G, y_G, z_G) in the domain D_G;
the first order derivative (μ', ν').

For the symmetry we save the position coordinates (x_G, y_G, z_G) in addition to (x_F, y_F, z_F). Because for each initial point we can search the intersection curve by the marching method in two opposite directions, each linked list has two pointers: one points to the head and the other points to the tail. For one direction we add data nodes through the pointer to the head and through the pointer to the tail for the other direction. In comparison of two nodes, we may compare the parameter values in both domains and their derivatives. Thus at a branch point if two preimages meet but do not have the same tangent vector, they will not be identified as the same point and the marching method will continue. For a better detection, we can save other higher order derivatives in the linked lists so that we can distinguish two different preimages even they have the same tangency somewhere.

As we are computing the surface-surface intersection numerically, a positive number as error tolerance should be provided in the comparison of the nodes. Also, in order to have an accurate detection of recomputations we should give a control on the maximum step size in adjusting step size for marching. In our implementation, we also provide a control on the minimum step size.

The process of marching along an intersection curve terminates only when (1) the boundary of a domain is reached; (2) a closed curve of intersection has been computed completely; (3) a degenerate point where four or more intersection curves intersect is encounted. Therefore the nodes for an intersection curve will be stored in the same linked list if the intersection curve does not contain degenerate points. When an intersection curve contains a degenerate point where four or more intersection curves intersect, very likely the nodes for it are stored in several linked lists. Therefore one should have an algorithm to connect these linked lists. Again, the stored information of the derivatives can be helpful in connecting the linked lists correctly.

For output display, a C^1 piecewise cubic polynomial curve can be easily constructed via the linked lists through Hermitian interpolation. Moreover, the information stored in the linked lists also can be used to find more accurate clipping on the boundary of the domains. For instance, we can use the parameter values and one of the derivatives to form a quadratic polynomial and then find the the intersection of the quadratic with the boundary line. The Newton-Raphson iteration on the boundary of the domains should be used for higher accuracy.

8. Example

As a conclusion of the paper, we show an example of two right cylinders of radius 1.

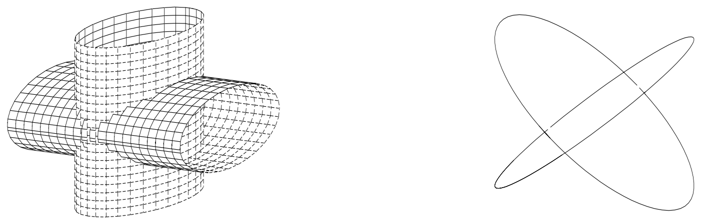

Fig. 2: surfaces and their intersection

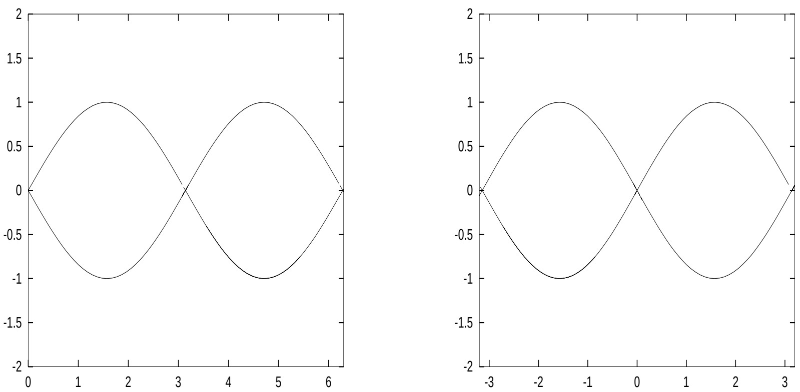

Fig. 3: preimages

From the graph one can see that the intersection of the two cylinders consists of two circle-like curves and there are two double points on the intersection. Since the two intersection curves do not have the same tangent line at each double point, they are also degenerate points. This is also observed from that the two cylinders have the same normal line at each double point.

Acknowledgments

I am grateful to the following people for the discussions and communications with them during my work on the project: Peter Alfeld, Chandrajit Bajaj, Carl de Boor, Scott Kersey, Ray Sarraga, Robert Schaback, ...

References

1. Bajaj, C., Hoffmann, C., Hopcroft, J., and Lynch, R., Tracing surface intersections, *Comput. Aided Geom. Des.*, **5**(1988), 285-308.
2. Bajaj, C., Xu, G., NURBS approximation of surface/surface intersection curves, *Advances in Computational Mathematics*, **2** (1994), 1-21.
3. Barnhill, R., Farin, G., Jordan, M., and Piper, B., Surface/Surface intersection, *Comput, Aided Geom. Des.*, **4**(1987), 3-16.
4. Barnhill, R. and Kersey, S., A marching method for parametric surface/surface intersection, *Comput. Aided Geom. Des.*, **7**(1990), 257-280.
5. Bürger, H. and Schaback, R., A parallel multistage method for surface/surface intersection, *Comput. Aided Geom. Des*, **10**(1993), 277-291.
6. Chen, J. and Ozsoy, T., Predictor-corrector type of intersection algorithm for C^2 parametric surfaces, *Comput. Aided Des.*, **20**(1988), 347-352.
7. Farin, G., An SSI Bibliography, *Geometry Processing for Design and Manufacturing*, R. Barnhill, eds., SIAM (1992), 205-207.
8. Faux, I. and Pratt, M., *Computational Geometry for Design and Manufacture*, Ellis Horwood Ltd, Chichester, UK, (1979).
9. Houghton, E., Emnett, R., Factor, J., and Sabharwal, C., Implementation of a divide-and-conquer method for intersection of parametric surfaces, *Comput. Aided Geom. Des.*, **2**(1985), 173-183.
10. Piegl, L., Constructive geometric approach to surface-surface intersection, *Geometry Processing for Design and Manufacturing*, R. Barnhill, eds, SIAM, (1992), 137-159.