A torus patch approximation approach for point projection on surfaces

Xiao-Ming Liu, Lei Yang, Jun-Hai Yong, He-Jin Gu, Jia-Guang Sun

A torus patch approximation approach for point projection on surfaces

Xiao-Ming Liu [a,b,c,d,e,*], Lei Yang [a,b,d,e], Jun-Hai Yong [a,d,e], He-Jin Gu [f], Jia-Guang Sun [a,b,d,e]

[a] School of Software, Tsinghua University, Beijing 100084, PR China
[b] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
[c] Command and Engineering College of Chemical Defense, Beijing 102205, PR China
[d] Key Laboratory for Information System Security, Ministry of Education, Beijing 100084, PR China
[e] Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, PR China
[f] Jiangxi Academy of Sciences, Nanchang 330029, PR China

ARTICLE INFO

Article history:
Received 13 December 2007
Received in revised form 22 December 2008
Accepted 12 January 2009
Available online 15 January 2009

Keywords:
Point projection
Torus patch approximation
Parametric surface

ABSTRACT

This paper proposes a second order geometric iteration algorithm for point projection and inversion on parametric surfaces. The iteration starts from an initial projection estimation. In each iteration, we construct a second order osculating torus patch to the parametric surface at the previous projection. Then we project the test point onto the torus patch to compute the next projection and its parameter. This iterative process is terminated when the parameter satisfies the required precision. Experiments demonstrate the convergence speed of our algorithm.

1. Introduction

The fundamental problem of projecting a test point onto a parametric surface to find the closest point on the surface as well as corresponding parameter value arises frequently in applications such as surface intersection (see Limaiem and Trochu, 1995), interactive object selection, boundary representation solid modeling, calculating tolerance in surface fitting, projection of a space curve onto a surface for surface curve design (see Pegna and Wolter, 1996) and shape registration (see Besl and McKay, 1992; Tucker and Kurfess, 2003; Pottmann et al., 2004; Hu and Wallner, 2005). When the test point is on the surface, the point projection problem turns to point inversion. Therefore, algorithms for point projection can also be applied to the point inversion problem.

Most of the early work on this problem uses the Newton–Raphson method to solve minimum distance equations. Mortenson (1985) gives a numerical approach. For the calculation of the minimum distance between the test point $\mathbf{P}$ and the surface, if $\mathbf{Q}$ is the point on the surface closest to $\mathbf{P}$, it must satisfy $(\mathbf{Q}-\mathbf{P})\times\mathbf{n}=\mathbf{0}$, where $\mathbf{n}$ is the normal vector of the surface at $\mathbf{Q}$. The roots of the equations are found by a numerical method. Limaiem and Trochu (1995) compute the orthogonal projection by constructing an auxiliary function and finding its zeros. Hartmann (1999) resolves the point projection problem by using a *normalform* equation of the surface and its first order derivatives. All of them use the Newton–Raphson method to find the roots of the equation set. Piegl and Tiller (1997) give a detailed and complete description of the Newton–Raphson iteration method to solve the problems of point projection and inversion.

To achieve reliable convergence, an important prerequisite for the Newton–Raphson method is a good initial value, which is hard to obtain due to the complexity of the NURBS surfaces (see Piegl and Tiller, 1997). The subdivision method is widely used to compute the closest point or to determine a good initial value for the Newton–Raphson method. Zhou

* Corresponding author at: Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China.
E-mail address: xm-liu06@mails.tsinghua.edu.cn (X.-M. Liu).

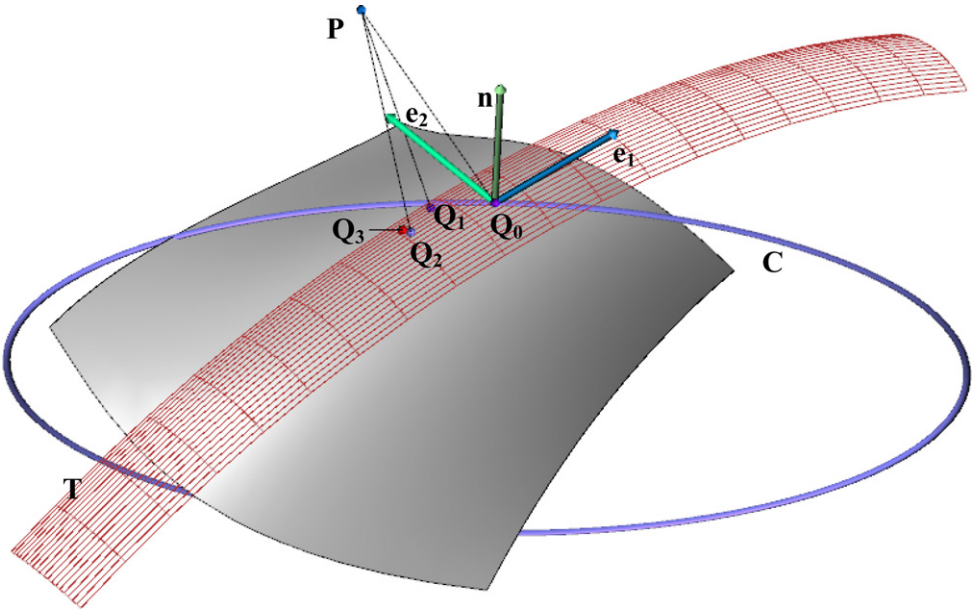

Fig. 1. Illustration of Hu and Wallner's algorithm and ours: Q_1 and Q_2 are projection points obtained by Hu and Wallner's algorithm and ours after the first iteration, respectively. Q_3 is the exact closest point.

et al. (1993) convert the problem into n polynomial equations with n variables expressed in the tensor product Bernstein basis. The solution is based on subdivision relying on the convex hull property of the n-dimensional Bernstein basis and minimization techniques. Dyllong and Luther (1999) also apply the subdivision techniques of a NURBS surface and solution of a nonlinear system. Piegl and Tiller (2001) decompose a NURBS surface into quadrilaterals, project the test point onto the closest quadrilateral, and then recover the parameter from that quadrilateral. Johnson and Cohen (1998) present an algorithm framework for minimum distance computation using the common Branch and Bound approach. Ma and Hewitt (2003) search for the initial value by the subdivision scheme and checking the relationship between the test point and the control point net of Bézier patches. But their elimination criterion may fail in some cases provided by Chen et al. (2007). Selimovic (2006) proposes a more practical exclusion criteria for the computation of the initial value.

Furthermore, the Newton–Raphson method quite often gives the wrong result or even fails to converge especially for points near the boundaries (see Piegl and Tiller, 2001). Even with a quite good initial value, the Newton–Raphson method still occasionally gives the wrong answer (see Ma and Hewitt, 2003). To avoid such a situation, Piegl and Tiller (2001) resort to the method of recovering the parameter from the corners of the closest quadrilateral instead of the Newton iteration. Ma and Hewitt (2003) apply the Newton–Raphson method on the quadrilateral which is a "flat enough" Bézier patch. The pure algebraic Newton–Raphson method is unsuitable for the problems of point projection and inversion.

Geometric iteration methods, which use only geometric information that is common to all possible parameterization, can accommodate arbitrary parameterizations in dealing with surfaces (see Hu and Wallner, 2005). The first order geometric iteration appears in Hoschek and Lasser (1993), Hartmann (1999). Hu and Wallner (2005) propose a second order geometric iteration method illustrated in Fig. 1, where **P** is a test point and $\mathbf{Q_0}$ is an initial estimate of the projection point. They create a normal curvature circle **C** coplanar with **P** and project **P** onto **C** to obtain the next projection point $\mathbf{Q_1}$.

In Hu and Wallner's (2005) method, the direction of the normal curvature is computed only by the first order information of the surface. And the local surface geometric information cannot be completely captured by the normal curvature circle. These characteristics of the method lower the stability and the convergence speed of the iteration. To overcome these limitations, we improve their method by replacing the circle of normal curvature with a second order osculating torus patch to the surface, which contains a complete set of local second order geometric information of the surface and is a more accurate local approximation of the surface.

In this paper, based on our torus patch approximation technique, we provide a novel second order geometric iteration algorithm for point projection and inversion. Given a test point **P**, a parametric surface and the initial parameter value of the roughly estimated projection $\mathbf{Q_0}$, as illustrated in Fig. 1, we need to compute the parameter of the precise projection. Our algorithm framework can be described in summary as follows.

(1) Compute the geometric information of the surface at the initial parameter value. The geometric information contains the position $\mathbf{Q_0}$, the normal vector **n**, the two principal curvatures and the corresponding principal vectors $\mathbf{e_1}$ and $\mathbf{e_2}$.

(2) Construct a torus patch **T** as a second order osculation to the surface at the initial point $\mathbf{Q_0}$ using the geometric information.

(3) Project the test point **P** onto the torus patch **T** and compute the new approximate parameter of the projection point $\mathbf{Q_2}$ on the original surface.

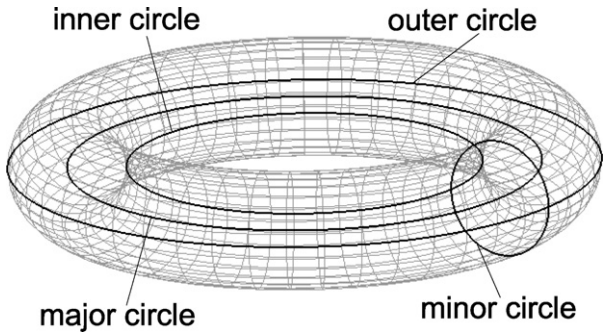

Fig. 2. A torus and circles associated with it.

(4) Use the new parameter as the initial value and repeat steps (1)–(3) until the parameter satisfies the required precision.

The rest of this paper is organized as follows. Section 2 presents the technique for surface approximation by the torus patch. In Section 3, the methods of point projection onto the torus patch and parameter inversion to the original surface are described. The experimental results including the evaluation of performance data are given in Section 4. Finally, Section 5 concludes the paper.

2. Local surface approximation by the torus patch

2.1. Properties of the torus

The torus can be defined as $\mathbf{T}(u, v) = ((R + r\cos v)\cos u, (R + r\cos v)\sin u, r\sin v)$, where $u, v \in [-\pi, \pi)$. In the following, we mainly refer to four characteristic circles related to the torus: major, minor, outer and inner circles, as shown in Fig. 2. We choose the torus as a local approximation of a parametric surface because the torus has the following important properties.

(1) Though the torus is a quartic surface, it is simple to compute point projection on it. Only two point-to-circle projections are required.

(2) A patch of a cylinder surface, a plane or a sphere can be regarded as the degenerated form of the torus patch.

(3) The principal curvatures and principal directions of a point on the outer (or inner) circle of the torus can be directly obtained without complicated computation.

2.2. Local approximation of a surface by the torus patch

Given a parametric surface $\mathbf{S}(u, v)$ and a parameter value (u_0, v_0) in its domain, we can compute the position $\mathbf{Q} = \mathbf{S}(u_0, v_0)$, the unit normal vector $\mathbf{n}$, the two principal curvatures κ_1, κ_2 of $\mathbf{S}$ at $\mathbf{Q}$, and the two corresponding unit principal vectors $\mathbf{e_1}$, $\mathbf{e_2}$.

We first preprocess the geometric information as follows. (1) If $|\kappa_1| > |\kappa_2|$, we exchange both κ_1, κ_2 and $\mathbf{e_1}$, $\mathbf{e_2}$. (2) If $\kappa_1 > 0$, we multiply $\mathbf{n}$, κ_1, and κ_2 with -1. Thus we assure that $|\kappa_1| \leqslant |\kappa_2|$ and $\kappa_1 \leqslant 0$.

After that, we create a torus $\mathbf{T}$: the center at $\mathbf{Q} + \mathbf{n}/\kappa_1$, the axis direction $\mathbf{e_2}$, the major and minor radii $-1/\kappa_1 + 1/\kappa_2$ and $1/|\kappa_2|$. It is clear that the torus has the identical principal directions and principal curvatures with $\mathbf{S}$ at $\mathbf{Q}$. So they have second order contact. Note the torus is unique since we have preprocessed the geometric information.

If $\mathbf{Q}$ is a parabolic point, i.e. $\kappa_1 = 0$ and $\kappa_2 \neq 0$, the torus $\mathbf{T}$ degenerates to a cylinder surface. For a planar point where $\kappa_1 = \kappa_2 = 0$, $\mathbf{T}$ degenerates to the tangent plane of $\mathbf{S}$ at $\mathbf{Q}$.

In our approach, we only use the torus patch, i.e. a portion of the torus centered at the osculating point Q, to approximate S. The patch is defined by limiting the parameter region to $[-0.5, 0.5] \times [-0.5, 0.5]$. We experimentally decide the size of this region in order to avoid slow convergence and false estimation. The major, minor, outer and inner circles of the whole torus are truncated accordingly to the corresponding circular arcs of the torus patch by the parameter region.

3. Point projection onto the torus patch and parameter inversion

3.1. Point projection onto the torus patch

For the torus patch described in Section 2.2, the point projection can be decomposed into two steps of point projection onto circular arcs: (1) project the test point on the outer (or inner) circular arc so as to get the minor circle where the projection is located; (2) project the test point on the minor circular arc to find the projection point we need.

3.2. Compute the original surface parameter from the projection on the torus

In each iteration, after obtaining the projection point $\mathbf{R}$ on the torus, we need to compute its original surface parameter (u_1, v_1).

Suppose that the original surface parameter of the osculating point $\mathbf{Q}$ is (u_0, v_0). We need to find a parameter (u_1, v_1) which satisfies that $\mathbf{S}(u_1, v_1) = \mathbf{R}$. Let $u_1 = u_0 + \Delta u$ and $v_1 = v_0 + \Delta v$, then the second order Taylor's expansion gives:

$$\mathbf{S_u} \cdot \Delta u + \mathbf{S_v} \cdot \Delta v + \left(\mathbf{S_{uu}} \cdot \Delta u^2 + 2\mathbf{S_{uv}} \cdot \Delta u \cdot \Delta v + \mathbf{S_{vv}} \cdot \Delta v^2\right)/2 = \mathbf{QR}, \tag{1}$$

where $\mathbf{S_u}, \mathbf{S_v}, \mathbf{S_{uu}}, \mathbf{S_{vv}}, \mathbf{S_{uv}}$ are the first order and second order partial derivatives of $\mathbf{S}$ at $\mathbf{Q}$. This is an overconstrained quadratic equation set that contains three equations with two variables Δu and Δv. In practice, the point on the torus patch is not exactly on the original surface, hence only an approximate parameter at the original surface can be obtained. We find the approximate solution of Eq. (1) by the Newton-type iteration method. The initial value for the iteration can be computed by omitting the second order terms in Eq. (1). Experiments show that the iteration converges fast enough for our purpose.

3.3. Dealing with parameters beyond the domain of the original surface

Sometimes the parameter obtained in Section 3.2 is beyond the domain of the original surface. We deal with them using a different approach from that of Piegl and Tiller (1997). Suppose that $\mathbf{p_0}(u_0, v_0)$ and $\mathbf{p_1}(u_1, v_1)$ are the parameters before the current iteration and after it, respectively. When $\mathbf{p_1}$ is out of the domain, we select the intersection point of the line segment $\mathbf{p_0p_1}$ and the boundaries of the domain as the new iteration parameter value. In the next iteration, if the result is still beyond the domain in the same direction, we apply the point projection method for parametric curves (see Piegl and Tiller, 1997) to find a projection point parameter on the boundary curve. To verify this result on the surface, we apply one more iteration on the surface. If the new parameter obtained is still outside the domain in the same direction, the algorithm terminates with the boundary projection as the final result. Otherwise, some more iterations on the surface are necessary.

4. Experimental results

There are three main criteria for evaluating point projection iteration methods.

(1) **Correctness.** In our experiments, if the distance between the computed projection and the true closest point satisfies a given precision, it is treated as a correct solution.

(2) **Speed of convergence.** We measure the convergence speed by two kinds of experimental data: the number of iterations and the CPU time. We record the average and the worst number of iterations in each computation.

(3) **Independence on the initial value.** The initial value has a significant impact on the correctness of the Newton-like iteration algorithms. Bad initial points which are far away from the true projection points often lead to wrong results. We do not elaborate on the methods of finding a good initial point, which are beyond the scope of this paper. The good initial values in the experiments of this paper are found by the subdivision method (see Selimovic, 2006; Ma and Hewitt, 2003), and the bad initial values are obtained by the intuitive method of grid-sampling the surface coarsely and picking the nearest sample point. The average and maximum differences between the initial parameter values and the parameters of the precise solutions are listed in Table 3.

This section compares our algorithm with Hu and Wallner (2005) second order algorithm. We apply the convergence criteria in Piegl and Tiller (1997), which is

$$\left\|\mathbf{S}(u_i, v_i) - \mathbf{P}\right\| \leqslant \epsilon_1, \tag{2}$$

$$\frac{|\langle \mathbf{S_u}(u_i, v_i), \mathbf{S}(u_i, v_i) - \mathbf{P}\rangle|}{\|\mathbf{S_u}(u_i, v_i)\| \cdot \|\mathbf{S}(u_i, v_i) - \mathbf{P}\|} \leqslant \epsilon_2, \qquad \frac{|\langle \mathbf{S_v}(u_i, v_i), \mathbf{S}(u_i, v_i) - \mathbf{P}\rangle|}{\|\mathbf{S_v}(u_i, v_i)\| \cdot \|\mathbf{S}(u_i, v_i) - \mathbf{P}\|} \leqslant \epsilon_2 \tag{3}$$

or

$$\left\|(u_{i+1} - u_i)\mathbf{S_u}(u_i, v_i) + (v_{i+1} - v_i)\mathbf{S_v}(u_i, v_i)\right\| \leqslant \epsilon_1, \tag{4}$$

where (u_i, v_i) is the parameter obtained at the ith iteration and ϵ_1, ϵ_2 are two zero tolerances of Euclidean distance and cosine, respectively. The iteration converges if Eqs. (2), (3), or (4) is satisfied. In our experiments, $\epsilon_1 = \epsilon_2 = 10^{-10}$ and the maximum number of iteration is set to be 10. The data in this paper are obtained using a personal computer with Core Duo 1.8 GHz CPU and 3 GB memory.

4.1. Test a single point projection onto Hu and Wallner (2005) surface

Example 1. We first test Example 3 of Hu and Wallner (2005). Table 1 lists the variations in parameter values during iteration and the CPU time with the test point (120, 10, 100) and the initial parameter (0.9, 0.6). Table 2 shows the case that the test point is (−120, 10, 100) and the initial parameter is set to be (0.1, 0.6).

In performance data of Table 1, Hu and Wallner (2005) method iterates 6 times while our method does 4. Our method consumes 50.4% CPU time of theirs. Table 2 shows that their method can not converge effectively after 10 iterations while our method iterates only 4 times and save 41.3% of the time consumed.

Table 1
Data for Example 1. The test point is (120, 10, 100), and $(u_0, v_0) = (0.9, 0.6)$.

Method	Step	1	2	3	4	5	6	CPU time (ms)
Hu and	Δu	−3.4e−02	−4.3e−03	3.8e−05	−5.1e−06	9.0e−08	−1.2e−08	0.751
Wallner's	Δv	−4.8e−02	6.5e−03	2.3e−04	7.3e−08	5.4e−07	2.0e−10	
Ours	Δu	−4.3e−02	3.9e−03	1.9e−05	5.1e−10			0.379
	Δv	−4.5e−02	3.8e−03	4.9e−06	1.6e−10			

Table 2
Data for Example 1. The test point is $(-120, 10, 100)$, and $(u_0, v_0) = (0.1, 0.6)$.

Method	Step	1	2	3	4	5	10	CPU time (ms)
Hu and	Δu	3.1e−02	−9.4e−03	7.0e−03	−4.9e−04	7.8e−04	−1.5e−07	0.569
Wallner's	Δv	2.9e−02	3.8e−02	1.4e−03	5.5e−03	5.1e−06	2.3e−06	
Ours	Δu	2.4e−02	5.1e−03	2.0e−04	3.0e−07			0.334
	Δv	7.2e−02	1.6e−03	5.7e−05	8.0e−08			

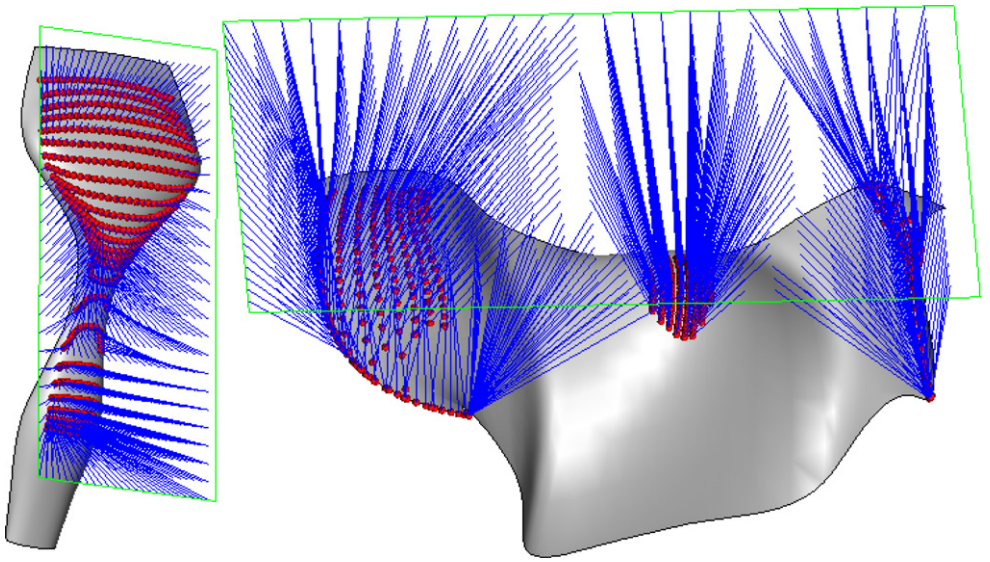

Fig. 3. Illustration of Examples 2 (left) and 3 (right).

Table 3
Data for Examples 2 and 3.

Example	Initial value error		Method	Correct solutions	Number of iterations		Total CPU time(ms)
	Average	Max			Average	Worst	
Example 2	0.00986	0.0467	H&W's	515	9.72	10	581
(bad initial value)			Ours	625	4.07	7	226
Example 2	0.000495	0.0145	H&W's	585	9.18	10	532
(good initial value)			Ours	625	3.01	5	186
Example 3	0.0131	0.0935	H&W's	561	6.28	10	247
(bad initial value)			Ours	625	3.18	9	150
Example 3	0.00350	0.0795	H&W's	595	6.03	10	260
(good initial value)			Ours	625	2.61	5	133

4.2. Points from a plane project onto surfaces

Examples 2 and 3. As illustrated in Fig. 3, an array of points including 25 columns and 25 rows, evenly sampled from a plane, are projected onto two surfaces, a spade model (Example 2) and an irregular B-spline surface (Example 3). The projection trajectories are visualized using solid lines. Table 3 lists the average and maximum differences between initial parameters and parameters of precise solutions, the number of correct solutions, the average and the worst number of

iterations in each computation and the total CPU time with bad or good initial values. The experiment of Example 3 shows that there are 212 projection points lying on the boundary of the surface.

Experimental result shows that our algorithm finds all correct solutions even with bad initial values while the successful ratios of Hu and Wallner (2005) method are 86.1% and 94.2% fed by the bad and the good initial values, respectively. The average number of iterations and the average CPU time of our method is only 1/2.43 and 1/2.33 that of theirs, respectively.

5. Conclusion

By taking full advantage of the second order geometric information of a parametric surface point, we propose a novel local surface approximation technique: torus patch approximation. The approximation torus patch and the original surface are second order osculating. Based on this, we provide a second order geometric iteration algorithm for point projection onto parametric surfaces. Experiments show that our algorithm converges at the given precision faster than the second order algorithm of Hu and Wallner (2005), which approximates the parametric surface with a circle of normal curvature. Moreover, our algorithm is less dependent on the choice of initial values.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. The research was supported by Chinese 973 Program (2004CB719400), the National Science Foundation of China (60625202, 60533070, 90715043), Chinese 863 Program (2007AA040401) and the INRIA/Tsinghua University Program (D 4748). The third author was supported by the project sponsored by the Fok Ying Tung Education Foundation (111070).

References

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

Chen, X.-D., Su, H., Yong, J.-H., Paul, J.-C., Sun, J.-G., 2007. A counterexample on point inversion and projection for NURBS curve. Computer Aided Geometric Design 24 (5), 302.

Dyllong, E., Luther, W., 1999. Distance calculation between a point and a NURBS surface. In: Laurent, P.J., Sablonniere, P., Schumaker, L. (Eds.), Curve and Surface Design, Saint-Malo, pp. 55–62.

Hartmann, E., 1999. On the curvature of curves and surfaces defined by normalforms. Computer Aided Geometric Design 16 (5), 355–376.

Hoschek, J., Lasser, D., 1993. Fundamentals of Computer Aided Geometric Design. A.K. Peters.

Hu, S.-M., Wallner, J., 2005. A second order algorithm for orthogonal projection onto curves and surfaces. Computer Aided Geometric Design 22 (3), 251–260.

Johnson, D.E., Cohen, E., 1998. A framework for efficient minimum distance computation. In: Proceedings of the 1998 IEEE International Conference on Robotics & Automation Leuven, Belgium, pp. 3678–3684.

Limaiem, A., Trochu, F., 1995. Geometric algorithms for the intersection of curves and surfaces. Computers and Graphics 19 (3), 391–403.

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

Mortenson, M.E., 1985. Geometric Modeling. Wiley. pp. 305–317.

Pegna, J., Wolter, F.-E., 1996. Surface curve design by orthogonal projection of space curves onto free-form surfaces. Journal of Mechanical Design 118 (1), 45–52.

Piegl, L.A., Tiller, W., 1997. The NURBS Book, second ed.. Springer-Verlag, Berlin, Heidelberg, New York.

Piegl, L.A., Tiller, W., 2001. Parameterization for surface fitting in reverse engineering. Computer-Aided Design 33 (8), 593–603.

Pottmann, H., Leopoldseder, S., Hofer, M., 2004. Registration without ICP. Computer Vision and Image Understanding 95 (1), 54–71.

Selimovic, I., 2006. Improved algorithms for the projection of points on NURBS curves and surfaces. Computer Aided Geometric Design 23 (5), 439–445.

Tucker, T.M., Kurfess, T.R., 2003. Newton methods for parametric surface registration. Part II. Experimental validation. Computer-Aided Design 35 (1), 115–120.

Zhou, J.M., Sherbrooke, E.C., Patrikalakis, N., 1993. Computation of stationary points of distance functions. Engineering with Computers 9 (4), 231–246.