A second order algorithm for orthogonal projection onto curves and surfaces

Shi-min Hu and Johannes Wallner

Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
shimin@tsinghua.edu.cn;
Institut für Diskrete Mathematik und Geometrie, TU Wien, Austria
wallner@geometrie.tuwien.ac.at.

Abstract

In this paper we analyze an algorithm which solves the point projection and the "inversion" problems for parametric curves and surfaces. It consists of a geometric second order iteration which converges faster than existing first order methods, and whose sensitivity to the choice of initial values is small. Applications include the ICP algorithm for shape registration.

1 Introduction and previous work

Projecting a point onto a parametric curve or surface in order to find the closest point (footpoint) and computing the parameter values of the projection (the *point inversion problem*) has attracted interest due to its importance in geometric modeling, computer graphics and computer vision, see e.g. Ma and Hewitt (2000) or Piegl and Tiller (2001). Both projection and inversion are essential for interactively selecting curves and surfaces (see Hu et al. (2001)), for construction and rendering of solid models with boundary representation, projecting of a space curve onto a surface for surface curve design (cf. Pegna and Wolter (1996)), and are also a key issues in the ICP (*iterative closest point*) algorithm for shape registration described in Besl and McKay (1992).

Several algorithms have been developed which solve these problems. Some depend on special properties of the objects to be projected onto, such as Mortenson (1985), who essentially finds the root of a polynomial using a Newton-Raphson method. Limaien and Trochu (1995) compute the orthogonal projection of a point onto parametric curves and surfaces by constructing an auxiliary function and finding its zeros. Hartmann (1999) proposes a first order algorithm for foot point computation by using a *normalform* (again, an auxiliary function) and its first derivatives.

Piegl and Tiller (2001) provide an algorithm for point projection on NURBS surfaces by decomposing a NURBS surface into quadrilaterals, projecting the test point onto the closest quadrilateral, and then recover the parameter from the closest quadrilateral. Ma and Hewitt (2000) present a practical algorithm for computing a good initial value for the Newton-Raphson method.

Apparently there are two key issues in the projection and inversion problems:
— computing a good initial value;
— and using a Newton-type or other iteration to improve the solution.

Naturally, all methods using derivatives of the target object have difficulties if the magnitude of oscillations of the target's surface is smaller than the test point's distance to the target. In that case, a zero order algorithm which consists in sampling the target and comparing distances is essentially the only way of finding a good initial value for a further iteration, or even for solving the problem at all. An example of this is given by Ma and Hewitt (2000), who find a good initial for NURBS curves or surfaces by subdividing into Bézier curves or surfaces and making use of relationship between control points and curve/surface (the control points being a very good sample of the target, including derivatives).

Algorithms which converge quickly usually employ first or second derivatives. It is natural to apply a Newton-type iteration. The sensitivity of this procedure to initial values is well known, as discussed e.g. in Ma and Hewitt (2000). On the other hand, applications like shape registration require fast algorithms for computing footpoints, as the projection part is actually the bottleneck of the entire computation, see Besl and McKay (1992) and Pottmann et al. (2004).

The main objective of this paper is to analyze a geometric iteration method, which solves the projection and inversion problems, and which has second order approximation properties, It uses only such second order information of the curve or surface under consideration which is geometric in the sense that it is common to all possible parameterizations. In that way a certain amount of the arbitrariness always present when parameterizations in dealing with surfaces is eliminated. We compute parameter values by projecting points to curvature circles and use the second order Taylor expansion of the curve or surface in order to compute parameter increments. Numerical evidence shows that this algorithm is robust and fast.

We will also show how such a second order geometric iteration is useful in shape registration of point clouds and improves the efficiency of the registration process.

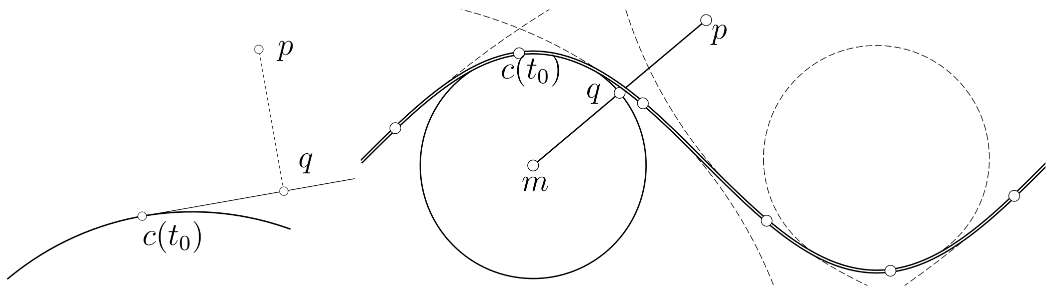

Fig. 1. Left: A first order algorithm for projection onto surfaces. Right: The curve $(t, \sin(t))$, its curvature circles, and a step of second order iteration.

2 Orthogonal projection onto a curve

Assume that $c(t)$ be a C^2 curve in n-dimensional Euclidean space $\mathbb{R}^n$ ($n \geq 2$), and p is a test point. A first order geometric iteration which computes the footpoint of p is the following: Projecting p onto the tangent of c at $t = t_0$ yields a point q expressible in terms of $c(t_0)$ and the derivative $c'(t_0)$ (see Fig. 1):

$$q = c(t_0) + \Delta t\, c'(t_0). \tag{1}$$

The scalar product of vectors $x, y \in \mathbb{R}^n$ will be denoted by $\langle x, y\rangle$, and the norm of a vector x by $\|x\|$. Then

$$\Delta t = \frac{\langle c'(t_0), q - c(t_0)\rangle}{\langle c'(t_0), c'(t_0)\rangle}. \tag{2}$$

We increment t_0 by Δt and repeat the above procedure until Δt is less than a given tolerance, or until the angle $\angle(c(t_0)qp)$ is close enough to $90°$. In this way we can compute the projection of p onto the curve in a simple way.

A geometric second order method is to replace the curve c by its curvature circle $\bar{c}$ at $t = t_0$. The curve's curvature will be denoted by the symbol κ. Recall that the curvature circle has radius $1/\kappa$ and lies on that side of the tangent where $c''(t_0)$ points to. The curvature is computed by the formula $\kappa = \mathrm{area}(c'(t_0), c''(t_0))/\|c'(t_0)\|^3$. Here $\mathrm{area}(x, y)$ denotes the area of the parallelogram spanned by the vectors x and y, possibly with sign. If $n = 2$, we have $\mathrm{area}(x, y) = \det(x, y)$. If $n = 3$, we use $\mathrm{area}(x, y) = \|x \times y\|$, and in general we have the formula $\mathrm{area}(x, y)^2 = \langle x, x\rangle\langle y, y\rangle - \langle x, y\rangle^2$. In any case the area has the properties that $\mathrm{area}(x + \lambda y, y) = \mathrm{area}(x, y + \lambda x) = \mathrm{area}(x, y)$ for all λ, and $\mathrm{area}(\lambda x, y) = \mathrm{area}(x, \lambda y) = \lambda\, \mathrm{area}(x, y)$ for all $\lambda \geq 0$. We compute the footpoint q of p on the curvature circle (or on the tangent, if κ happens to be zero; see Fig. 1).

We assume for the moment the curvature circle $\bar{c}$ to be parameterized such that it has the same Taylor polynomial as the curve c. We use the symbol $o(\Delta t^2)$ for any vector-valued or real-valued function $r(\Delta t)$ such that $\lim_{\Delta t \to 0} \frac{1}{\Delta t^2} r(\Delta t) = 0$. Then we have

$$\begin{aligned} q &= \bar{c}(t_0 + \Delta t) = c(t_0 + \Delta t) + o(\Delta t^2) && (3) \\ &= c(t_0) + \Delta t c'(t_0) + \frac{\Delta t^2}{2} c''(t_0) + o(\Delta t^2) && (4) \end{aligned}$$

In $\mathbb{R}^2$, we may take the determinant of the previous equation with either $c'(t_0)$. We get

$$\det(q - c(t_0), c''(t_0)) = \Delta t \det(c'(t_0), c''(t_0)) + o(\Delta t^2),$$

which yields the formula

$$\Delta t + o(\Delta t^2) = \frac{\det(q - c(t_0), c''(t_0))}{\det(c'(t_0), c''(t_0))} = \frac{1}{\kappa \|c'\|^3} \det(q - c(t_0), c''(t_0)). \qquad (5)$$

From this, the parameter increment Δt may be computed easily by simply disregarding the remainder term $o(\Delta t)^2$.

In the general case (n may now be greater than 2), we compute area$(q - c(t_0), c'(t_0))$ with q from Equ. (3). We get

$$\begin{aligned} \operatorname{area}(c', q - c(t_0)) &= \frac{\Delta t^2}{2} \operatorname{area}(c', c'') + o(\Delta t^2) && (6) \\ \Longrightarrow \ \Delta t^2 &\approx 2 \frac{\operatorname{area}(c', q - c(t_0))}{\operatorname{area}(c', c'')} = 2 \frac{\operatorname{area}(c', q - c(t_0))}{\kappa \|c'\|^3}. && (7) \end{aligned}$$

The sign of Δt is chosen according to

$$\operatorname{sign}(\Delta t) = \operatorname{sign}\langle c'(t_0), q - c(t_0) \rangle. \qquad (8)$$

These equations now can be used to compute the parameter increment Δt. Iteration yields a second order algorithm for computing the footpoint of x onto the curve together with the parameter value of the footpoint.

For $n = 2$, both formulas (5) and (7) are equivalent in the limit $t \to t_0$, but the second one, which only includes the first derivative vector and the curvature, leads to a more stable iteration.

This algorithm also solves the *inversion problem* which means computing the parameter value t for a point which is known to lie on the curve.

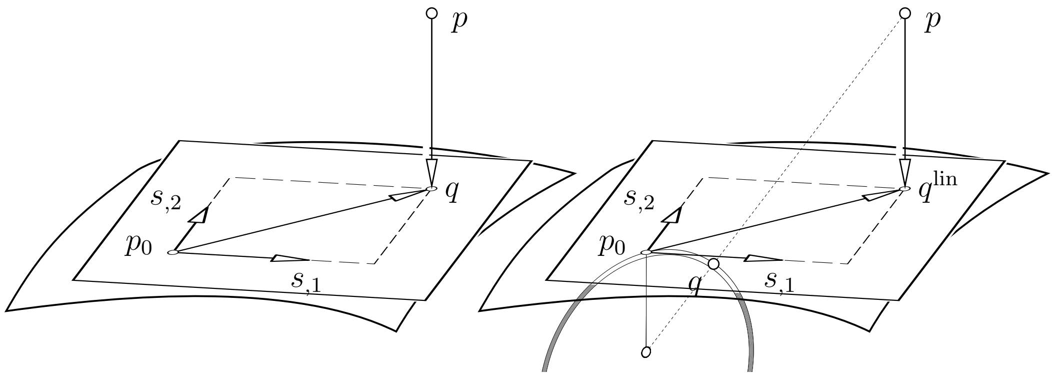

Fig. 2. Illustration of first order (left) and second order (right) geometric iteration for surfaces

3 Orthogonal projection onto a surface

We extend the geometric iteration described above to surfaces $s(u^1, u^2)$ in $\mathbb{R}^3$. Partial derivatives with respect to the parameters u^1 and u^2 will be denoted by $s_{,1}$, $s_{,2}$, $s_{,11}$, and so on. The coefficients of the first fundamental form are given by $g_{ij} = \langle s_{,i}, s_{,j} \rangle$, the unit normal vector field by $n = (s_{,1} \times s_{,2})/\sqrt{\det(g_{jk})}$, and the coefficients of the second fundamental form by $h_{jk} = \langle s_{,jk}, n \rangle$. We assume that s is regular, i.e., $\{s_{,1}, s_{,2}\}$ is linearly independent, so $\det(g_{jk}) = \operatorname{area}(s_{,1}, s_{,2})^2 \neq 0$.

Projecting a point p onto a surface is done as follows. We assume that we already have an initial guess $p_0 = s(u_0^1, u_0^2)$, and that we find q by projecting p onto the tangent plane at p_0 (see Fig. 2):

$$q - p_0 = s_{,1}.\Delta u^1 + s_{,2}.\Delta u^2. \tag{9}$$

By multiplying with $s_{,i}$ $(i = 1, 2)$ we get

$$\langle s_{,1}, s_{,1} \rangle \Delta u^1 + \langle s_{,2}, s_{,1} \rangle \Delta u^2 = \langle q - p_0, s_{,1} \rangle, \tag{10}$$
$$\langle s_{,1}, s_{,2} \rangle \Delta u^1 + \langle s_{,2}, s_{,2} \rangle \Delta u^2 = \langle q - p_0, s_{,2} \rangle, \tag{11}$$

so $\Delta u^1, \Delta u^2$ can be computed as solution of a regular system of linear equations, with coefficient matrix (g_{jk}). We update u_0^1, u_0^2 by adding Δu^1, Δu^2. This first-order geometric iteration appears in Hartmann (1999), Hoschek and Lasser (1993), and Hu et al. (2000).

In order to improve efficiency, we would like to propose the following approach of geometric approximation by normal curvature. Any vector $x - p_0$ can be expressed as a linear combination of the tangent vectors $s_{,1}$, $s_{,2}$ and the normal vector n at p_0:

$$x - p_0 = \lambda^1 s_{,1} + \lambda^2 s_{,2} + \nu n. \tag{12}$$

The normal curvature of the tangent vector $\lambda^1 s_{,1} + \lambda^2 s_{,2}$ can be computed via

$$\kappa_n = (\sum_{i,j=1}^{2} h_{ij}\lambda^i\lambda^j) \cdot (\sum_{i,j=1}^{2} g_{ij}\lambda^i\lambda^j)^{-1}. \tag{13}$$

We consider a planar section of the given surface with the plane which contains p_0, the normal vector n, and the point x. It has some parameterization $c(t)$ which we actually will not need, and which has the property that $c(0) = p_0$, and that its tangent vector is given by

$$c'(0) = \lambda^1 s_{,1} + \lambda^2 s_{,2}. \tag{14}$$

Its radius of curvature is given by $1/\kappa_n$. The circle of curvature is contained in the plane mentioned above, and has the center $p_0 + n/\kappa_n$.

We project the point x onto the circle of curvature, which yields the point q. The orthogonal projection of x onto the surface is now approximated by $c(\Delta t)$, with Δt computed according to Equ. (7) (c' is taken from (14)). The sign of Δt is that of the scalar product $\langle c'(0), q - p_0\rangle$ We now update u^1 and u^2 according to

$$u^i \longrightarrow u^i + \Delta u^i, \quad \Delta u^i = \lambda^i \Delta t. \tag{15}$$

The procedure is repeated again, with $s(u^1, u^2)$ as new initial point, until the desired accuracy criteria are met.

4 Examples

This section shows numerical evidence concerning the behaviour of the first and the second order geometric algorithms discussed above.

Example 1. We consider the curve $c(t) = (t, \sin t)$, depicted in Fig. 1 together with 7 evenly distributed curvature circles. Table 1 shows the results of geometric iteration. Here, the initial parameter t_0 is estimated by comparing the distance between the test point and 7 uniformly sampled points with parameters $t = 2k\pi/7$, $k = 0, 1, \ldots, 7$. The experimental data show the second order algorithm has good convergence, and but the convergence of the first order algorithm is sometimes very slow.

$x=(1,0.8)$ $t_0=0.898$						
step	1	2	3	4	5	6
Δt_1	8.1e-02	2.7e-03	5.7e-05	1.2e-06	2.3e-08	0.0
Δt_2	8.4e-02	1.8e-04	6.0e-10	0.0	$t=0.982347$	
$x=(2,2)$, $t_0=1.795$						
step	1	2	3	4	5	6
Δt_1	-2.2e-02	2.1e-02	-2.0e-02	1.9e-02	-1.8e-02	1.7e-02
Δt_2	-1.1e-02	2.5e-05	1.0e-10	0.0	$t=1.783812$	

Table 1
Step sizes Δt_1 and Δt_2 in Example 1 for the first and second order algorithms.

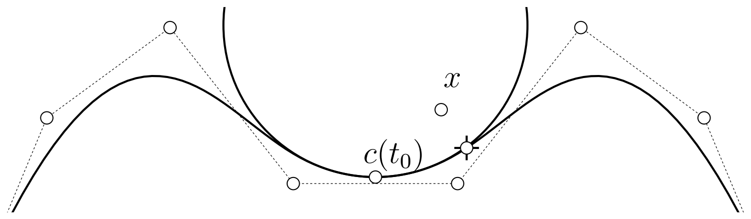

Fig. 3. Illustration of geometric iteration for the B-Spline curve

Example 2. We consider the order B-Spline curve $c(t)=\sum_{i=0}^{n} b_i B_i^4(t)$ with the knot list $(0,0,0,0,.2,.4,.6,.8,1,1,1,1,1)$ and the control points $(100,100)$, $(140,196)$, $(200,240)$, $(260,164)$, $(340,164)$, $(400,240)$, $(460,196)$, $(500,100)$. An initial parameter guess for the footpoint q of a point x may be obtained by finding the control point nearest to x (see reference Ma and Hewitt (2000)), or by computing distances to a sample of points $c(t_i)$.

Fig. 3 shows the initial point x to be projected, together with the first guess $c(t_0)$ for a footpoint and the curvature circle there. The "+" sign denotes the result of one step of iteration.

Table 2 compares the first and second order algorithms, the faster convergence of the latter being clearly visible. Table 3 shows the robustness of the second order algorithm with respect to the choice of an initial value t_0. The solution in this case is given by $t=0.6223419238$.

Example 3. In order to give also a surface example, we consider the 4×4 B-Spline surface $S(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{6} p_{ij} B_{i,4}(u) B_{j,4},(v)$, whose control points are given by $(-236, -197, -22)$, $(-206, -117, -22)$, $(-216, -27, 8)$, $(-246, 62, -22)$; $(-156, -177, 8)$, $(-176, -97, 38)$, $(-157, 20, 126)$, $(-186, 142, 8)$; $(-86, -157, 8)$, $(-138, -113, -146)$, $(-104, 14, -60)$, $(-96, 102, 8)$; $(-6, -197, -22)$, $(-47, -96, -33)$, $(25, 32, 95)$, $(-6, 102, 8)$; $(74, -177, 8)$, $(34, -75, 147)$, $(86, 97, 105)$, $(54, 142, 8)$; $(124, -157, 8)$, $(198, -31, 63)$, $(64, 31, 154)$, $(144, 102, 8)$; $(204, -197, -22)$, $(234, -77, 8)$, $(214, -7, 8)$,

$x = (381, 252)$, $t_0 = 0.75$,						
step	1	2	3	4	5	6
Δt_1	3.2e-02	−2.3e-02	−1.8e-02	−1.4e-02	1.1e-02	−8.8e-03
Δt_2	2.1e-02	−1.2e-03	−4.1e-06	0.0	$t = 0.7695140103$	
$x = (332, 200)$, $t_0 = 0.5$						
step	1	2	3	4	5	6
Δt_1	8.5e-02	2.9e-02	6.9e-03	1.3e-03	2.2e-04	3.9e-05
Δt_2	1.2e-01	1.2e-03	−3.8e-06	0.0	$t = 0.6223419238$	

Table 2
Stepsizes Δ_1 and Δt_2 for the first and second order algorithms together with the solution t corresponding to Example 2.

step	1	2	3	4	5	6
$t_0 = 0.30$	1.4e-01	1.7e-01	1.1e-02	−2.9e-04	−2.0e-07	0.0
$t_0 = 0.40$	2.0e-01	2.0e-02	−1.1e-03	2.7e-06	0.0	
$t_0 = 0.50$	1.2e-01	1.2e-03	−3.8e-06	0.0		
$t_0 = 0.60$	2.4e-02	−1.4e-03	−5.1e-06	1.0e-10		
$t_0 = 0.70$	−6.6e-02	−1.1e-02	−2.9e-04	−2.0e-07	0.0	
$t_0 = 0.80$	−1.1e-01	−6.1e-02	−9.0e-03	−1.9e-04	8.8e-08	0.0

Table 3
Convergence rate in terms of Δt of the second order algorithm for different choices of initial value t_0 ($x = (332, 200)$ in Example 2).

first order algorithm.						
step	1	2	3	4	5	6
Δu^1	−3.8e-02	−2.4e-06	−3.8e-04	−1.4e-04	−4.3e-05	1.2e-05
Δu^2	−5.4e-02	1.4e-02	−2.6e-03	−5.1e-04	−1.1e-04	2.5e-05
second order algorithm.						
step	1	2	3	4	5	6
Δu^1	−3.4e-02	−4.3e-03	3.8e-05	−5.1e-06	9.0e-08	−1.2e-08
Δu^2	−4.8e-02	6.5e-03	2.3e-04	7.3e-08	5.4e-07	2.0e-10

Table 4
Data for Example 3. ($x = (120, 10, 100)$, $(u_0^1, u_0^2) = (0.9, 0.6)$).

(239, 102, −22). The knot lists are $(0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1)$ for u and $(0, 0, 0, 0, 1, 1, 1, 1)$ for v. An initial estimate for the projection has been obtained by means of the control polyhedron. Table 4 shows experimental results with the test point $(120, 10, 100)$ and initial parameter $(0.9, 0.6)$. Table 5 shows a case were the first order algorithm fails. The test point is $(-120, 10, 100)$, the initial parameter was set to $(0.1, 0.6)$.

first order algorithm.						
step	1	2	3	4	5	10
Δu^1	6.9e-02	$-$7.3e-02	9.6e-02	$-$1.0e-01	1.2e-01	$-$1.2e-01
Δu^2	6.3e-02	2.5e-02	$-$7.0e-02	7.6e-02	$-$9.7e-02	1.0e-01
second order algorithm.						
step	1	2	3	4	5	10
Δu^1	3.1e-02	$-$9.4e-03	7.0e-03	$-$4.9e-04	7.8e-04	$-$1.5e-07
Δu^2	2.9e-02	3.8e-02	1.4e-03	5.5e-03	5.1e-06	2.3e-06

Table 5
Experimental data for Example 3. Test point is $(-120, 10, 100)$, and $(u_0^1, u_0^2) = (0.1, 0.6)$.

5 Application to the ICP algorithm for shape registration

The shape registration problem for a given design model and a set of data points (i.e., a point cloud approximating the shape of the design model) amounts to finding a rigid body motion such that its application to the design model minimizes an appropriately defined distance of the design model from the point cloud.

A well known standard algorithm to solve such a registration problem is the iterative closest point (ICP) algorithm of Besl and McKay (1992). It usually consists of two steps. First, for points x_i in the cloud the respective closest points y_i on the model are computed. Second, a motion m is found such that $\sum \mathrm{dist}(m(x_i), y_i)$ is minimized. The first step is the most time consuming part of the algorithm and has to implemented efficiently, see e.g. Pottmann et al. (2004). Other registration algorithms such as the Newton method of Tucker and Kurfess (2003) or the squared distance function method of Pottmann et al. (2004) depend on computing closest points, i.e., computing orthogonal projections.

Second order algorithms as those discussed in this paper have some properties which make them suitable for projection and for accelerating the ICP algorithm. First, if x_i and x_j are close together, we may always use the footpoint y_i of x_i as an initial value for the computation of y_j; and when iterating the computation of the motion m, we may use the footpoints of the previous step as an initial value for the current one. While this is true for most projection algorithms, it is probably even more so for ours, as it is rather insensitive with respect to initial values.

Depending on the oscillatory behaviour of the surface in question (which would be tame for many applications, a fact usually known beforehand), we might expect, due to numerical observations, that about 95% of the total parameter

j	0	1	2	3	4	5	6	7	8
$E(j)$	1.04	3.5e-1	2.1e-1	1.1e-1	6.0e-2	1.3e-2	4.3e-4	1.7e-6	2.3e-11
$\frac{E(j)}{E(j-1)}$		0.34	0.60	0.53	0.54	0.22	3.2e-2	3.7e-3	1.4e-5

Table 6
Convergence rate of the registration process illustrated in Fig. 4. Object size: $\approx 2.0 \times 0.8 \times 0.4$. The qantity $E(j)$ is defined as $\sqrt{(\sum_i (x_i - y_i)^2)/k}$, where j is the number of iterations.

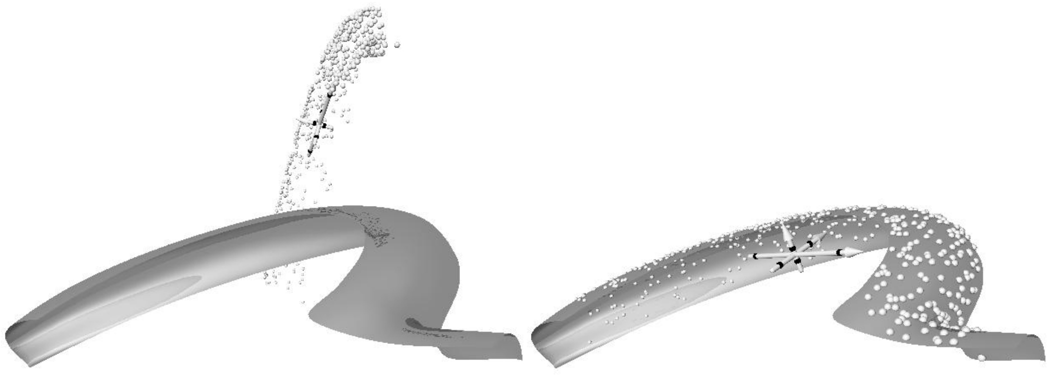

Fig. 4. Left: Before registration. Right: After registration

increment during projection is achieved in the first step. This means that it would be sufficient to perform just one step of the projection algorithm in order to ensure convergence of the ICP algorithm. Fig. 4 shows a registration example. It has been computed using the algorithm of Pottmann et al. (2004), which is faster than the traditional ICP method, as documented by Table 6.

Conclusion

This paper investigates point projection and point inversion on parametric curves and surfaces by using curvature information. Experimental results show that the algorithms under consideration are robust and efficient. Applications to shape registration are discussed.

Acknowledgements

This work was supported by the Austrian Science Foundation (Grant No. P15911) during the first author's stay in Vienna; and further by the Natural Science Foundation of China (Project number 60273012, 60225016, 60333010) and the National Basic Research Project of China (Project Number 2002CB

312101). The authors would like to thank the reviewers for their helpful comments.

References

Besl, P. J., McKay, N. D., 1992. A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. 14, 239–256.

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

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

Hu, S.-M., Li, Y.-F., Ju, T., Zhu, X., 2001. Modifying the shape of NURBS surfaces with geometric constraints. Computer Aided Design 33, 903–912.

Hu, S.-M., Sun, J.-G., Jin, T.-G., Wang, G.-Z., 2000. Computing the parameter of points on NURBS curves and surfaces via moving affine frame method [Chinese]. J. Software 11, 49–53.

Limaien, A., Trochu, F., 1995. Geometric algorithms for the intersection of curves and surfaces. Computers & Graphics 19, 391–403.

Ma, Y. L., Hewitt, W. T., 2000. Point inversion and projection for NURBS curve and surface: Control polygon approach. Computer Aided Geometric Design 20, 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. J. Mech. Design 118, 45–52.

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

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

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