Approximation by Ruled Surfaces

Horng-Yang Chen and Helmut Pottmann
Institut für Geometrie, Technische Universität Wien
Wiedner Hauptstraße 8–10, A–1040 Wien, Austria

Abstract

Given a surface or scattered data points from a surface in 3–space, we show how to approximate the given data by a ruled surface in tensor product B–spline representation. This leads us to a general framework for approximation in line space by local mappings from the Klein quadric to Euclidean 4–space. The presented algorithm for approximation by ruled surfaces possesses applications in architectural design, reverse engineering, wire electric discharge machining and NC milling.

keywords: computer aided design, NC milling, wire EDM, reverse engineering, line geometry, ruled surface, surface approximation.

1 Introduction

Ruled surfaces are formed by a one–parameter set of lines and have been investigated extensively in classical geometry [6, 7]. Because of their simple generation, these surfaces arise in a variety of applications including CAD, architectural design [1], wire electric discharge machining (EDM) [10, 13] and NC milling with a cylindrical cutter. In the latter case and in wire EDM, when considering the thickness of the cutting wire, the axis of the moving tool runs on a ruled surface Φ, whereas the tool itself generates under peripheral milling or wire EDM an offset of Φ (for properties of ruled surface offsets, see [9]). For all these applications the following question is interesting: Given a surface, how well may it be approximated by a ruled surface? In case of the existence of a sufficiently close fit, the ruled surface (or an offset of it in case of NC milling with a cylindrical cutter) may replace the original design in order to guarantee simplified production or higher accuracy in manufacturing. This is an example of kinematics–driven geometric modeling in the sense of Q. J. Ge [4].

Thinking of applications like architectural design, only point sets on the surface such as laser scanner data taken from a 3D model of the shape may be available. Therefore, we also allow sufficiently dense scattered data points as input in our algorithm. Thus, the present paper also contributes to reverse engineering of geometric models [12].

In this article we will describe a method how to create an approximating ruled surface in tensor product B–spline representation. The algorithm consists of 3 steps:

1. Find a discrete system of rulings close to the given surface.
2. Construct a ruled tensor product B–spline surface approximating the system of rulings.
3. Optional: Improve the approximating ruled surface by known surface approximation techniques, which are a combination of least square approximation with functional minimization and parameter correction [2, 8].

These steps will be explained in the next section, followed by several examples. The second step is treated within a general framework for interpolation and approximation of line segments. It also leads to a concept for *approximation in line space* with help of local mappings of the Klein quadric into Euclidean 4–space.

2 The approximation algorithm

2.1 Find a discrete system of rulings

Input of the algorithm is either a parametric or implicit representation of a surface $\Phi \subset \mathbb{R}^3$. If scattered data points are available only, they are first fitted by a surface. Assuming sufficient density of the data points, even a piecewise linear model is sufficient for our task. In fact, it is advisable to compute such a model in any case in order to speed up the computations in the following first step of the algorithm.

Let us first treat the case, where there exists a vector e_3 and an angle $0 < \gamma_0 < \pi/2$ such that all surface normals form an angle $0 < \gamma < \gamma_0$ with e_3. Then, we may put the z–axis of a Cartesian coordinate system (x, y, z) parallel to e_3 and the surface may be written as graph of a bivariate function $(x, y, f(x, y))$ over some domain D' in the x, y–plane π_1. We do not need to compute f explicitly. From here on, elements in π_1 will be indicated by primes.

Consider all lines in π_1, which intersect D' in some segment l'. Disjoint segments in the same line are treated separately. We now assign each of these segments l' exactly to one line segment l in 3-space, namely the fitted line segment of the curve traced out by the plane strip going through l' and perpendicular to π_1 (Fig.1). The fitted line segment l is computed by minimizing the least square error.

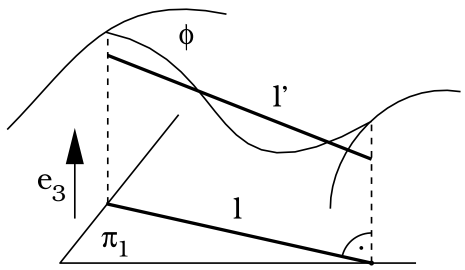

Figure 1: Corresponding fitted line of a line segment

Furthermore we assign each line segment in D' the mean square error of the corresponding fitted line, which reflects how close it will be to the given surface. Because of the bijection of a possible ruling in 3-space and its topview in the plane π_1, it is sufficient to guide our construction via the system in π_1.

First we choose a central point d'_0 of the domain D' and compute among the line segments through d'_0 the one with the least error. This segment l'_0 is the starting segment for the following marching algorithm, whose initialization also requires the choice of an initial marching direction orthogonal to l'_0. For constructing the segment next to l'_i, consider all line segments passing through the point d'_{i+1} on the normal ray through the midpoint c'_i of the previous line segment l'_i in distance d, which do not intersect the previous line segment (see Fig.2).

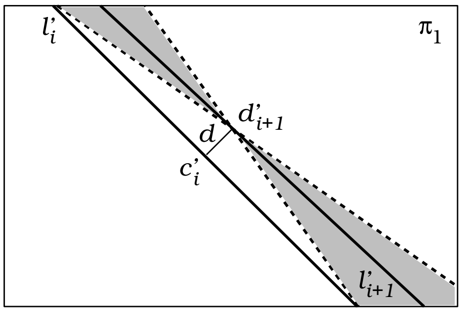

Figure 2: The shaded area indicates valid positions for the next ruling.

Choose the line segment l'_{i+1} with the least error among these for the next ruling; if there are more than one minimum, take the line whose angle is smallest to the previous line segment. Repeating this procedure will yield the system of rulings. The value d determines how dense the constructed system will be. Nonintersection of two following rulings will help to avoid fold overs in the approximating surface.

In case that the system of rulings does not (approximately) close, the algorithm will have to be performed a second time, starting at l'_0 in the other direction. We are dealing with ruled surface strips only, and thus the topology of the given surface can either be that of a disc (open strip), a disc with one hole (orientable closed strip) and a Möbius strip. We exclude the latter case due to practical irrelevance. However, also for a closed orientable surface, the algorithm needs not deliver a nearly closing system if the deviation from a ruled surface is too large. Then, marching in both directions and final averaging of the obtained sequences is recommended.

To overcome the restriction to a surface which is a function graph over some plane, we just have to move the reference frame as well during the marching algorithm. A good position for the moving plane π_1 is the (true or estimated) tangent plane at the midpoint of the previous line segment.

2.2 Approximation of a system of lines by a ruled surface

The result of the first step is a system of line segments $l_0, \ldots, l_N$, approximating the given data and respecting its boundary. In the second stage, this system shall be approximated by a ruled surface strip

$$x(u, v) = c(u) + vg(u), \ v \in [-1, 1], u \in [a, b]. \tag{1}$$

In the sequel, we call $c(u)$ its *central curve* and $g(u)$ its *directrix*.

The present approximation problem is acting on the set L of *oriented line segments* in Euclidean 3–space $\mathbb{R}^3$. Therefore, it seems appropriate to think about the structure of this set and a simple computational approach. The following model is different from that proposed by Ge and Ravani [5] and exhibits high compatibility with current geometric design techniques [8].

An oriented line segment l can be captured by the ordered pair (p, q) of its endpoints, which naturally defines a mapping into real affine 6–space $\mathbb{R}^6 = \mathbb{R}^3 \times \mathbb{R}^3$,

$$\sigma : L \to \mathbb{R}^6, \quad (p, q) \mapsto X = (x_1, \ldots, x_6) = (p_x, p_y, p_z, q_x, q_y, q_z). \tag{2}$$

Points as segments in $\mathbb{R}^3$ are mapped to points in the 3–dimensional subspace $P: \ x_1 = x_4, x_2 = x_5, x_3 = x_6$. The change of orientation in all segments is seen as affine reflection at P.

To get a useful measure for the distance between two line segments $l_i = (p_i, q_i)$, consider the similarity mapping between the two segments,

$$(1 - \lambda)p_1 + \lambda q_1 \mapsto (1 - \lambda)p_2 + \lambda q_2, \ \lambda \in [0, 1].$$

Connecting associated points leads a system of line segments on a bilinear patch (on a hyperbolic paraboloid or plane) connecting l_1 and l_2. It seems to be appropriate, to define the L^2 norm of the Euclidean distance function of these segments (muliplied by 3 for convenience) as their squared distance,

$$\begin{aligned} d(l_1, l_2)^2 : &= 3 \int_0^1 [(1 - \lambda)(p_1 - p_2) + \lambda(q_1 - q_2)]^2 d\lambda \\ &= [(p_1 - p_2)^2 + (q_1 - q_2)^2 + (p_1 - p_2) \cdot (q_1 - q_2)]. \end{aligned} \tag{3}$$

The thereby defined distance is simply the distance of the two points $L_i := \sigma(l_i) \in \mathbb{R}^6$ in the *Euclidean metric* induced by the positive definite quadratic form

$$\langle X, X\rangle = x_1^2 + \ldots + x_6^2 + x_1x_4 + x_2x_5 + x_3x_6. \tag{4}$$

Thus, we have derived the following result that performs a transition principle from interpolation and approximation problems on line segments in $\mathbb{R}^3$ to standard point–based interpolation and approximation in Euclidean 6–space.

Lemma 1 *The mapping σ, defined in (2), provides a distance preserving bijection between oriented line segments in Euclidean 3–space and points in real Euclidean 6–space E^6, defined as $\mathbb{R}^6$ with the positive definite quadratic form (4). Here, the distance between line segments in 3–space is defined in (3).*

The present approximation problem thus reduces to a curve approximation problem. Note that a B–spline curve of degree k in 6–space corresponds to a B–spline tensor product patch of degree $(1, k)$ in $\mathbb{R}^3$. Standard curve approximation techniques including parameter correction can be applied [8], but clearly Euclidean measures have to be taken with the inner product according to (4).

To achieve a better behaviour of the approximating surface, the least squares fit is often combined with the minimization of a regularization functional. For ruled surfaces in the form (1), a parameter dependent, but usually good choice is the thin plate spline functional

$$F(x) = \int_a^b \int_{-1}^1 (x_{uu}^2 + 2x_{uv}^2 + x_{vv}^2) dv du. \tag{5}$$

Inserting (1), we find

$$F(x) = 2\int_a^b c_{uu}^2 \; du + \frac{2}{3}\int_a^b (g_{uu}^2 + 6g_u^2) du, \tag{6}$$

i.e. a decomposition into well–known functionals applied to the central curve and to the directrix curve, respectively.

Lemma 2 *For a ruled surface strip (1) the thin plate spline functional (5) decomposes into the cubic spline functional applied to the central curve and a tension spline functional applied to the directrix curve according to (6).*

The solutions of the two univariate varational problems are well–known in CAGD [8]. Under interpolation and appropriate boundary or closure conditions, the solution surface has a cubic spline as central curve and a tension spline as directrix curve; approximation leads to the corresponding smoothing spline schemes. However, the tension spline has exponential terms and thus we would not get a B–spline surface as solution. To approximate the tension spline behaviour sufficiently well by a cubic C^2 spline, we therefore will introduce additional knots between those knots corresponding to the given line segments l_i. At least when interpolating the segments l_i this should be done.

We have expressed the thin plate spline functional $F(x)$ with help of the surface representation (1) in order to realize its thereby easily seen decomposition into well–known functionals and in order to show how the solution of the associated variational problem $F \to$ min looks like. Moreover, it indicates which spline spaces to use for approximation. However, for using $F(x)$ as regularization functional in the approximation as outlined above, the computation is as follows: Represent the ruled surface with its boundary curves $p(u), q(u)$ as $(1-v)p(u) + vq(u),\ v \in [0,1]$, i.e. set $c(u) = (p(u)+q(u))/2$ and $g(u) = (p(u)-q(u))/2$. Inserting cubic B–spline representations over a fixed knot vector for p and q amounts to the choice of a cubic B–spline curve $X(u)$ in $\mathbb{R}^6$,

$$X(u) = (p(u), q(u)) = \sum_{i=0}^{M} N_i^3(u) X_i.$$

The thin plate bending energy $F(x(u))$ is a quadratic function $F^*(X_0, \dots, X_M)$ in the unknown control points $X_i = (p_i, q_i)$ for the curve $X(u)$ in $\mathbb{R}^6$ (control points p_i, q_i of the boundary curves to the surface $x(u,v)$ in $\mathbb{R}^3$). We now minimize the quadratic function in X_i,

$$G(X_0, \dots, X_M) := \sum_{i=0}^{N} \langle L_i - X(u_i), L_i - X(u_i) \rangle + \lambda F^*(X_0, \dots, X_M). \tag{7}$$

Both the parameters u_i to the given line segments l_i ($L_i = \sigma(l_i)$) and the parameter $\lambda > 0$ that determines the influence of the smoothing term have to be chosen prior to the minimization. Since the parameter choice has usually a heavy influence on the result, one will start with an initial guess for the parameters (for example, chord length parameterization for the polygon $L_0, \dots, L_M$ in $\mathbb{R}^6$ in the metric (4)). Then, we keep the knots of the spline fixed and use Hoschek's method [8] in order to iteratively change the parameters u_i until the error vectors are nearly orthogonal to the solution within the metric (4).

Remark 1. The endpoints of the input line segments l_i to the second step are reflecting the boundary behaviour of the input surface and this needs not be coherent with the boundary induced by the rectangular parameter domain of the approximating ruled surface (1). Therefore, prior to application of the second step, the following *segment alignment* should be performed. Compute a reference polygon $(r_0, \dots,)$, starting at the midpoint of any segment, say l_0, and computing the foot r_{i+1} of the normal from r_i onto the line carrying l_{i+1} (Fig. 3).

The polygon vertices r_i need not lie on the segments l_i and for a closed surface the polygon is in general not closed. Write the midpoints m_i of l_i in the form $m_i = r_i + \lambda_i e_i$, with $e_i := (p_i - q_i)/\|p_i - q_i\|$ and apply any smoothing technique

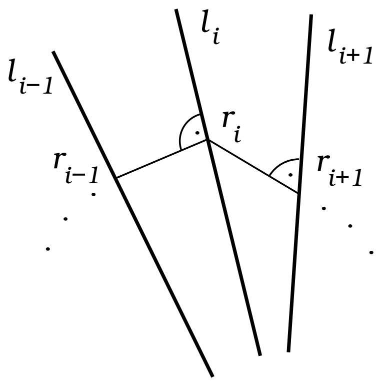

Figure 3: Reference point sequence for segment alignment

(e.g. discrete smoothing spline) to the function values λ_i, associated to abscissae $i = 0, \ldots, N$. This delivers a new sequence μ_i and corrected midpoints $c_i = r_i + \mu_i e_i$ of our ruling segments. Analogously, a smoothed sequence ν_i for the spline segment lengths is obtained. We finally use line segments l_i^* with endpoints $c_i \pm (\nu_i + \delta)e_i$ as input to step 2. Here, the value δ is added to cover the whole data set; the final surface is then represented as a *trimmed B–spline surface* [8].

2.3 Final adaption

The approximating ruled surface from step 2 has been constructed with the system of approximating line segments from the first step. Moreover, these rulings have been derived with error distances in z–direction of the moving frame instead of orthogonal distances to the given surface.

We therefore recommend a fine tune of the approximant using well–known surface approximation techniques. These are a combination of least squares fits including a regularization functional such as (5) and parameter correction [2, 8]. Good results with these methods require a good initial fit, which we have constructed in the first two steps.

Remark 2. In reverse engineering, one should be able to handle *outliers* in the data, which may either be large measurement errors or data points belonging to geometrically different parts of the model. The least squares methods in steps 1 and 3 would be too sensitive to outliers. Therefore, one may use a robust regression method to compute an initial estimate (e.g. an estimate to a least median of squares solution in the sense of the introduced metric [11]). This approximant may then be refined by so–called *M–estimators* [11]. These are iterative weighted least squares schemes that reject outliers or reduce their influence by assigning appropriate weights to them. Actually, all discussed applications can benefit from this modification.

2.4 Example

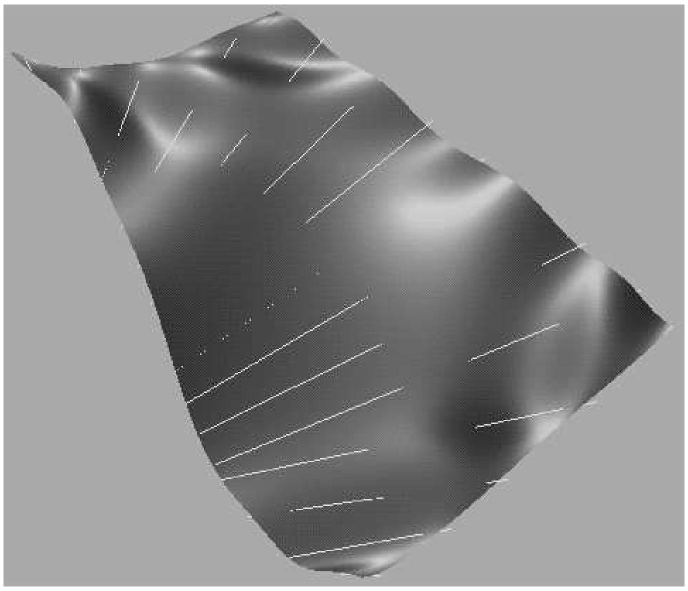

Figure 4: S(x,y) with close discrete set of ruling

We implemented our algorithm and will illustrate the 3 steps of it by an example. Given the surface

$$S(x,y) = (x, y, c(u) + vg(u) + r(x,y)), \text{(see Fig.4)},$$

$$\begin{aligned} c(u) &= 3(u^2 - 16)\sin(u), \\ g(u) &= (4 - u^2)\cos(u), \\ r(x,y) &= 0.5(\cos(0.3x^2)\sin(0.6xy)), \end{aligned}$$

$$u = \frac{x}{6+y}, v = y, x \in [-4.1, 4.1], y \in [-3.1, 3.1].$$

$S(x,y)$ is not a ruled surface, since it contains the term $r(x,y)$. Without $r(x,y)$ $S(x,y)$ would be a part of a ruled surface with centrix $c(u)$ and directrix $g(u)$. Furthermore $S(x,y)$ is a graph of a bivariate function over a rectangular domain D' in the x,y-plane.

Following step one the program computes the line segment through the central point of D' with least error and continues with the marching algorithm on both sides. Since it is not possible to do the computations considering every single line segment in D', this procedure is done on a discrete set evenly distributed among the set of candidates for the next ruling. The chosen discrete system of rulings is displayed in Fig. 4, Fig. 5 shows its topview in D'.

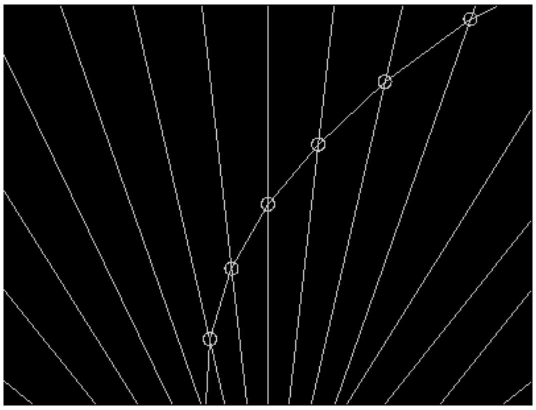

Figure 5: Marching algorithm, reference point sequence for segment alignment, topview

Coordinating the line segments in D' by its angle to the x-axis and its distance to the midpoint, we can visualize the "error-function" of the line segments with the angle as the x-axis, the distance as the y-axis and the error as the z-axis. The marching algorithm finds a "path through the valley" from one border (line segments with distance equal 0) to the other (line segments, which barely intersect D') see Fig. 6.

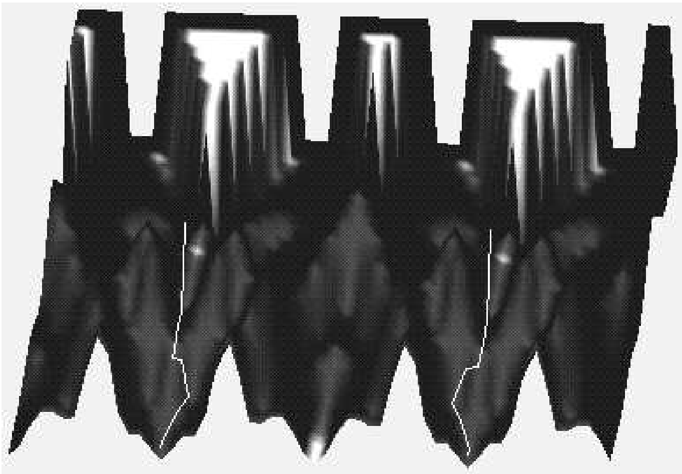

Figure 6: Path through the valley

The boundary of the discrete set of ruling is induced by the rectangular shape of D'. Following Remark 1 the program performs a segment alignment and calculates 2 sequences c_i and d_i of boundary points which reflects the boundary behavior of the input surface. Fig. 5 shows the topview of the midpoint sequence $\frac{c_i+d_i}{2}$ (the topviews c_i, d_i themselves are outside of D').

The sequence $L_i = (c_i, d_i) \in \mathbb{R}^6$ is used for solving the minimization of the quadratic function G. This yields the control points X_i for the boundary curves in $\mathbb{R}^6$ and the desired ruled surface in tensor product B-spline representation in $\mathbb{R}^3$. Fig. 7 presents the resulting approximating surface of this algorithm (elevated above the given surface).

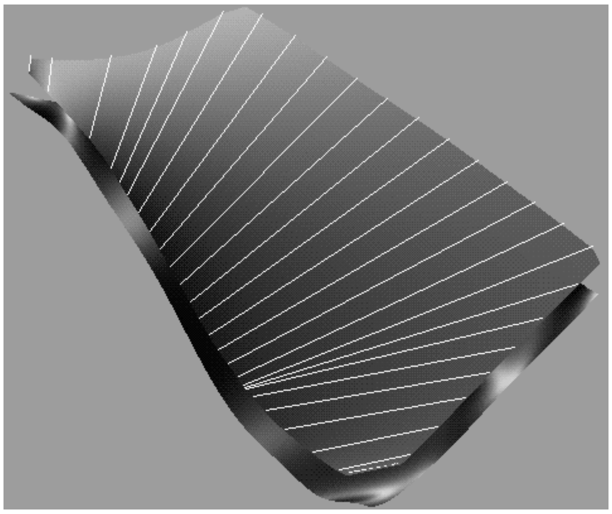

Figure 7: Approximating ruled surface

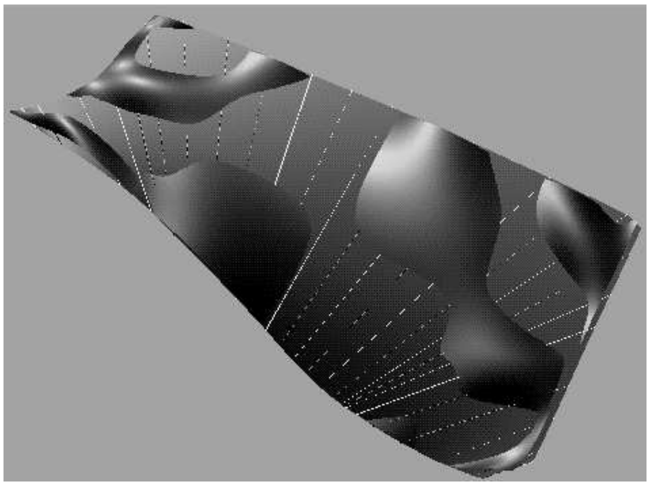

Figure 8: Deviation of the calculated ruled surface from the given surface

Fig. 8 shows how well the calculated ruled surface is smoothing out the deviation of the given surface.

3 Approximation in line space

Our method for approximation in the space of line segments leads us quite naturally to the following concept for *approximation in the space $\underline{L}$ of lines in* $\mathbb{R}^3$.

To understand the nature of the problem, let us briefly review a few facts from line geometry [6]. We use homogeneous Cartesian coordinates $(x_0, \ldots, x_3)$ in projectively extended $\mathbb{R}^3$ with $x = x_1/x_0, y = x_2/x_0, z = x_3/x_0$ for points not at infinity. Then, one defines the six homogeneous *Plücker coordinates* of a line l through points $(x_0 : \cdots : x_3)$ and $(y_0 : \cdots : y_3)$ as

$$(l_1, \ldots, l_6) := (l_{01}, l_{02}, l_{03}, l_{23}, l_{31}, l_{12}) \text{ with } l_{ij} := x_i y_j - x_j y_i. \tag{8}$$

These coordinates do not depend on the choice of the two points on l and are related by the *Plücker identity*

$$l_1 l_4 + l_2 l_5 + l_3 l_6 = 0. \tag{9}$$

There is a bijective map between ordered, homogeneous 6–tupels $(l_1, \ldots, l_6) \neq (0, \ldots, 0)$ of real numbers and lines in real projective 3–space P^3. Therefore, one may view the 6 Plücker coordinates of a line l as homogeneous coordinates of a point $l\gamma$ in real projective 5–space P^5. The thereby defined *Klein mapping* γ provides a bijection between the set of lines in P^3 and the set of points in a quadric $\Omega \subset P^5$ with equation (9), usually referred to as *Klein quadric.* We see that projective line space has the structure of a quadric in P^5.

If we work in $\mathbb{R}^3$ and thereby rule out lines at infinity, we remove a 2–dimensional plane (γ–image of the lines at infinity) from the Klein quadric. Thus, approximation in the set $\underline{L}$ of lines in $\mathbb{R}^3$ means approximation in the Klein quadric with a plane Π being removed. Unfortunately the resulting set $\Omega' := \Omega \setminus \Pi$ does not have the structure of an affine space. However, it is well–known that removal of a cut with a tangential hyperplane Γ at some point C of a quadric gives the structure of an affine space. The mapping to an affine space is then realized by *stereographic projection* with center C. From line geometry, we know that the points of a tangential cut of Ω at point $C \in \Omega$ are the Klein images of all lines in 3–space which intersect the line $c = C\gamma^{-1}$. Since we work in $\mathbb{R}^3$ and thus neglect lines at infinity, *the following operation introduces an affine structure into $\underline{L}$: remove all lines which intersect some line at infinity c.* We take this line at infinity c as that to the x, y–plane and thus remove the set $\underline{L}'$ of lines orthogonal to the z–axis to get $\underline{L}^o := \underline{L} \setminus \underline{L}'$.

The realization of the corresponding stereographic projection with center C is really simple: A line $l \in \underline{L}^o$ may be defined by its intersection point $p = (a_1, a_2, 0)$ with $\pi_1 : \ z = 0$ and its intersection $q = (a_3, a_4, 1)$ with $\pi_2 : \ z = 1$. The mapping

$$\sigma^o : \ l \in \underline{L} \mapsto (a_1, a_2, a_3, a_4) \in \mathbb{R}^4 \tag{10}$$

from the set $\underline{L}^o$ onto real affine 4–space is the resctriction of σ (2) onto the set of line segments with $p \in \pi_1,\ q \in \pi_2$. It decribes a stereographic projection of the Klein quadric: The Plücker coordinates of l and c are

$$l\gamma = (a_3 - a_1, a_4 - a_2, 1, a_2, -a_1, a_1 a_4 - a_2 a_3),\ \ c\gamma = (0, \ldots, 0, 1).$$

Embedding $\mathbb{R}^4$ into P^5 by $(a_1, a_2, a_3, a_4) \mapsto (a_3 - a_1, a_4 - a_2, 1, a_2, -a_1, 0)$, these points are collinear with $l\gamma$ and $c\gamma$.

For approximation we need a distance measure between two lines $l, \bar{l}$. In practice, the distance within some area of interest will be important. Placing this area between parallel planes π_1 and π_2, we may just use the distance of the line segments $(p, q), (\bar{p}, \bar{q})$ in the sense of (3),

$$\begin{aligned} d(l, \bar{l})^2 &= (p - \bar{p})^2 + (q - \bar{q})^2 + (p - \bar{p}) \cdot (q - \bar{q}) \\ &= \sum_{i=1}^{4} (a_i - \bar{a}_i)^2 + (a_1 - \bar{a}_1)(a_3 - \bar{a}_3) + (a_2 - \bar{a}_2)(a_4 - \bar{a}_4). \end{aligned} \tag{11}$$

This is the distance of their image points $l\sigma^o, \bar{l}\sigma^o \in \mathbb{R}^4$ in a *Euclidean metric in* $\mathbb{R}^4$, defined by

$$\langle X, X \rangle = x_1^2 + \ldots + x_4^2 + x_1 x_3 + x_2 x_4. \tag{12}$$

For the definition of $d(l, \bar{l})$ one integrates squared distances between $l, \bar{l}$ measured in parallel planes between π_1, π_2. These distances differ from orthogonal distances to l by a factor between 1 and $1/\cos\varphi$, if φ is the angle between l and the z–axis. At least for lines whose angle with the z–axis does not exceed some tolerance dependent value $\gamma_0 < \pi/2$, (11) is a useful distance measure. Its behaviour is a counterpart to the well–known fact, that the distance distortion for stereographic projection of a sphere in $\mathbb{R}^3$ increases with the distance from the antipodal point of the projection center.

Theorem 3 *Consider two parallel planes π_1, π_2 in $\mathbb{R}^3$ and the set $\underline{L}^o$ of all lines which are not parallel to them. Then intersection of any line in $\underline{L}^o$ with π_1, π_2 gives a pair (p, q) of points, which may be considered as point in real affine 4–space $\mathbb{R}^4$. This mapping from $\underline{L}^o$ onto $\mathbb{R}^4$ can be interpreted as stereographic projection of the Klein quadric. The image space $\mathbb{R}^4$ can be endowed with a Euclidean metric (in an adapted coordinate system given by (12)), which corresponds to the deviation of the lines within the parallel strip bounded by planes π_1, π_2.*

Let us illustrate the obtained *transfer principle from approximation in line space to approximation in Euclidean 4–space* at hand of two examples. A more detailed investigation is one of our topics for future research and will be presented elsewhere.

Example 1 We consider the problem of *scattered data interpolation and approximation for functions defined on line space.* Let a finite set of lines l_i (data lines) and associated real numbers f_i, obtained by some measurement or computation, be given. We would like to construct a function F, which is defined on all lines l within some domain D of interest (for example, lines with a maximum distance d to a fixed point) and exactly or approximately satisfies $F(l_i) = f_i$. One approach is the following. Consider a centrally symmetric covering of the unit sphere $\Sigma \in \mathbb{R}^3$ by circular caps $\Gamma_i,\ i = 1, \ldots, m$ with rotational axes a_i and spherical radii ρ_i. For each axis a_i, let $\underline{L}_i$ be the set of lines that form an angle $< \rho_i$ with a_i and lie in the domain of interest. With two parallel planes that are orthogonal to a_i and enclose the domain of interest, we perform the mapping into $\mathbb{R}^4$. There, the images of data lines $l_k \in \underline{L}_i$ are data points with associated function values. Using the corresponding metric, they can be interpolated or approximated by any method which works in $\mathbb{R}^4$, for example radial basis functions [8]. This gives a partial solution function F_i defined on any line in $\underline{L}_i$. Finally one just has to combine these partial solutions into a single one. Using the angles to a_i, the corresponding weight functions can be the same as for interpolation over the sphere Σ [3].

Example 2 We return to step 2 of our algorithm for *ruled surface approximation*. To approximate the discrete set of lines l_i, we consider their normalized direction vectors as points in the unit sphere and cover them by a circular cap with axis a and radius $\rho < \pi/2$; if this is not possible one has to split the data into subsets and later merge partial solutions. As in the previous example one can now transform the approximation problem into a curve approximation problem in $\mathbb{R}^4$ and use standard techniques there. This approach has the advantage that we do not have to care about segment alignment as discussed in remark 1.

Acknowledgements

This work has been supported by grant No. P11357-MAT of the Austrian Science Foundation. H. Pottmann would like to thank Bahram Ravani for providing funding through a grant from the California Department of Transportation and for offering a stimulating environment at the Department of Mechanical and Aeronautical Engineering of the University of California at Davis, where part of this research has been performed.

References

[1] H. Brauner and W. Kickinger, *Baugeometrie 1* (Bauverlag, 1977).

[2] U. Dietz, Erzugung glatter Flächen aus Messpunkten, Techn. Report 1717, Dept. of Mathematics, TH Darmstadt, 1995.

[3] M. Eck and H. Pottmann, Modified multiquadric methods for scattered data interpolation over a sphere, *Computer Aided Geometric Design* 7 (1990) 313–321.

[4] Q. J. Ge, Kinematics–driven geometric modeling: a framework for simultaneous NC tool–path generation and sculptured surface design, *Proc. IEEE Intl. Conf. on Robotics and Automation* (Minneapolis, MN, 1996) 1819–1824.

[5] Q. J. Ge and B. Ravani, On representation and interpolation of line segments for computer aided geometric design, *ASME Design Automation Conf.* , vol. 96-1 (1994) 191–198.

[6] V. Hlavaty, *Differential Line Geometry* (P. Nordhoff Ltd., Groningen, 1953).

[7] J. Hoschek, *Liniengeometrie* (Bibliograph. Institut, Zürich, 1971).

[8] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design* (A. K. Peters, Wellesley, MA, 1993).

[9] H. Pottmann, W. Lü and B. Ravani, Rational ruled surfaces and their offsets, *Graphical Models and Image Processing* 58 (1996) 544–554.

[10] B. Ravani and J. W. Wang, Computer aided design of line constructs, *ASME J. Mechanical Design* 113 (1991) 363–371.

[11] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection* (Wiley, New York, 1987).

[12] T. Varady, R. R. Martin and J. Cox, Reverse engineering of geometric models – an introduction, *Computer Aided Design* 29 (1997) 255–268.

[13] M. Yang and E. Lee, NC verification for wire–EDM using an R–map, *Computer Aided Design* 28 (1996) 733–740.