Eurographics Symposium on Geometry Processing (2003)
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

Approximate implicitization via curve fitting

E. Wurm, B. Jüttler

Department of Applied Geometry, Johannes Kepler University, Linz, Austria

Abstract

We discuss methods for fitting implicitly defined (e.g. piecewise algebraic) curves to scattered data, which may contain problematic regions, such as edges, cusps or vertices. As the main idea, we construct a bivariate function, whose zero contour approximates a given set of points, and whose gradient field simultaneously approximates an estimated normal field. The coefficients of the implicit representation are found by solving a system of linear equations. In order to allow for problematic input data, we introduce a criterion for detecting points close to possible singularities. Using this criterion we split the data into segments and develop methods for propagating the orientation of the normals globally. Furthermore we present a simple fallback strategy, that can be used when the process of orientation propagation fails. The method has been shown to work successfully

Categories and Subject Descriptors (according to ACM CCS): G.1.2 [Approximation]: Approximation of surfaces and contours, Spline and piecewise polynomial approximation; J.6 [Computer-Aided Engineering]: ; I.3.5 [Computational Geometry and Object Modelling]: Curve, surface, solid, and object representations

1. Introduction

Curves and surfaces in geometric modelling can be described by parametric and implicit representations. Currently, most applications rely on parametric representations, since they offer a number of advantages, such as simple sampling techniques (generation of an approximating triangulation for visualisation). However, parametric representations introduce a parameterisation of the geometry, which is often artificial. For instance, in order to fit curves or surfaces to scattered data, one has to associate certain parameter values to the data. Often, this determines the shape of the solution. Using implicit representations, it is possible to avoid this problematic parametrisation process.

In order to exploit the potential of implicit representations, methods for conversion to and from implicit representations are needed. In this paper we discuss the process of implicitization. Various exact methods [5, 6, 7, 10], such as resultants, Groebner bases, moving curves and surfaces[15], have been considered. Recently, some approximative methods[4, 8] have emerged.

Our approach for implicitization uses least squares curve/surface fitting of scattered data[1, 2, 14, 16]. More precisely we minimise the squared algebraic distances[14] of a set of given points from the zero contour of an unknown bivariate respective trivariate (piecewise) polynomial function F. In order to be able to solve this quadratic minimisation problem for the unknown coefficients of F, an additional constraint has to be introduced. As an alternative to the standard approach of a 'normalisation' in the coefficient space, we use estimated normals[12, 13], as additional information on the shape of the given data.

In this paper we show how reliable estimates can be generated for curve fitting allowing problematic data (e.g. data which may contain singular points). The first sections of this paper discuss normal estimation, detection of problematic points, natural segmentation of the data and the process of normal orientation propagation. After these preparations we present how the estimated normals are used in the task of curve fitting.

2. Normal estimation and ambiguity criterion

As the first step of our algorithm we estimate a unit normal $\mathbf{n}_p$ at every given point $\mathbf{p}$. This is done by computing the 'local line of regression' of the k closest neighbour points $\mathbf{q}_i$, similar to Hoppe et al.[11] and Gopi et al.[9]. Let $\mathbf{c}_p = \frac{1}{k}\sum_{i=1}^{k}\mathbf{q}_i$ be their centroid, and $\mathbf{v}_i = \mathbf{q}_i - \mathbf{c}_p$. The deviation of the points $\mathbf{q}_i$ from a line with the normal vector $\mathbf{n}$ through $\mathbf{c}_p$

equals

$$S(\mathbf{n}) = \sum_{i=1}^{k} (\mathbf{v}_i \cdot \mathbf{n}^\top)^2. \quad (1)$$

In order to minimise (1) subject to $\|\mathbf{n}\| = 1$, we have to solve the eigenvalue problem

$$M\mathbf{n} = \lambda\mathbf{n} \quad \text{where} \quad M = \sum_{i=1}^{i} \mathbf{v}_i \mathbf{v}_i^\top.$$

The matrix M is symmetric and positive semidefinite. Let $0 \le \lambda_1 \le \lambda_2$ be the eigenvalues and $\mathbf{e}_1, \mathbf{e}_2$ denote the associated eigenvectors. It is easy to see that $\mathbf{e}_1$ / $\mathbf{e}_2$ minimises / maximises $S(\mathbf{n})$, and the minimum / maximum of $S(\mathbf{n})$ is equal λ_1 / λ_2. In the neighbourhood of self-intersections, cusps or sharp features, λ_1 and λ_2 have approximatively the same value. Then the quality of the fit is approximatively the same for all lines passing through $\mathbf{c}_p$. We use this property to introduce a criteria that detects these features. Let $0 < C < 1$ be a user defined constant. For each point $\mathbf{p}$ we check whether the inequality

$$\frac{\lambda_1}{\lambda_2} \le C \quad (2)$$

holds or not. If a point passes this test, we define it's type to be 1, and set $\mathbf{n}_p$ as $\mathbf{e}_1$. Otherwise the type of $\mathbf{p}$ is set to be 0.

Recently, mor sophisticated methods for feature detection, based on local moment analysis, have been developed by Clarenz et. al.[3].

3. Segmentation of the data

In order to construct segments of points of the same type we now apply a simple region growing process. Segments of type 0 will represent regions around singularities, sharp features and other problematic parts of the input data. In figure 2 an example for the result of this process is shown, based on the estimated normals given in figure 1. Different segments can be distinguished by randomly assigned colours.

Starting with an arbitrary point $\mathbf{s}$, we consider a fixed number l of closest neighbours $\mathbf{q}_i$, $i = 1,..,l$. Among them we take the subset of points that have of the same type as $\mathbf{s}$. The closest one of these points is added to the segment. For segments already consisting of several points, we consider the union of the l closest neighbours, that additionally have the same type as $\mathbf{s}$. Again we add the point, which has the least distance towards the segment. The process stops, when no such point exists. The type of a segment is then determined by the type of it's points.

Simultaneously while building the segments we propagate the orientation of the estimated normals $\mathbf{n}_p$ within the segments of type 1. This is simply done by checking the scalar product of the normal of the point which is to be added and the normal of the closest point that already belongs to the segment. If this product is negative, then the orientation of the normal is needs to be swapped

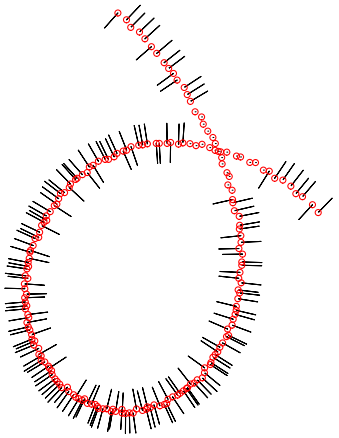

Figure 1: *Estimated normals*

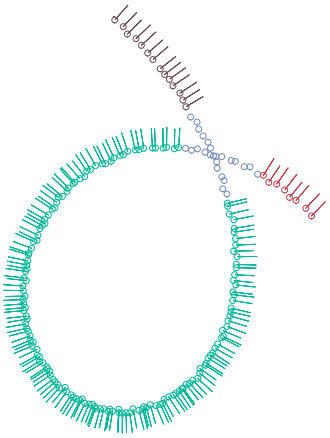

Figure 2: *Generating segments*

3.1. Global orientation propagation

Finally we have to ensure globally correct orientations for the segments. Segments of type 0 need not to be checked, since no estimates $\mathbf{n}_p$ are taken into account. However these segments can be used to guarantee a correct orientation of their neighbour segments, as follows.

Let S^* be a segment of type 0. For each neighbouring segment S_k we choose the closest point $\mathbf{r}_k \in S_k$ as representative. These points $\mathbf{r}_k$ can be arranged in a circular order around the centroid $\mathbf{c}$ of S^*. To ensure that the corresponding normals $\mathbf{n}_k$ are oriented correctly, we consider the rotated vectors $\mathbf{n}_k^\perp$ (by 90°), and require that the scalar products $(\mathbf{r}_k - \mathbf{c}) \cdot \mathbf{n}_k^\perp$ have alternating signs. If the orientation of a vector $\mathbf{n}_k$ has to be swapped, then we apply this to all estimates in the segment S_k. Again, a global suitable normal field is found by a region-growing process. Figure 3 shows a possible global orientation, for the segments of figure 2.

If the orientation of a segment, whose orientation has al-

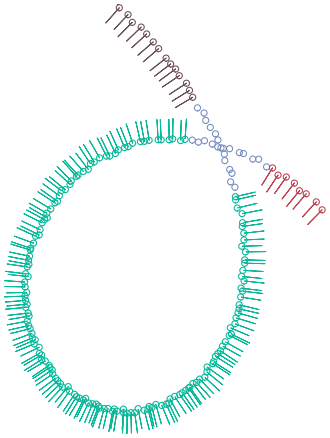

Figure 3: *Global orientation propagation*

ready been checked, is forced to change its orientation, an error is likely. Later we describe a simple but powerful fallback strategy to resolve this problem, see section 4.7.

Once an initial segment has been selected, we expect the algorithm to check all segments. If one needs to select more than one initial segment in order to reach all segments, then the data consists of more than one separated part. In this case, we consider a separate process of approximative implicitization for each part of the curve.

Note that the algorithm implicitly assumes that the number of neighbouring segments is always even. Otherwise the 'alternating sign' criterion does not work. In figure 5 such segments occur at the midpoints of the sides of the great square. A way to address this problem would be to introduce pairs of opposite neighbours and an additional 'phantom' segment opposing to the last free neighbour segment.

4. Approximative implicitization by piecewise polynomials

4.1. Outline

In order to describe the shape of a given set of points by the zero contour of a bivariate tensor-product function F, we minimise the sum of the squared algebraic distances of the data with respect to the zero contour of the function F. This leads to a quadratic minimisation problem for the unknown coefficients of F. Since the zero vector (the null function) belongs to the space of solutions, an additional constraint has to be introduced. Often this constraint is chosen as a 'normalisation' in the coefficient space[14, 16].

Using a linear constraint (e.g. one coefficient is chosen to be equal 1), the solution of the minimisation problem is found via a system of linear equations, but the solution is not geometrically invariant[14]. In the case of (geometrically invariant) quadratic constraints the solution is found as eigenvector related to the minimal eigenvalue of a certain matrix[16].

The additional constraint used by our method is geometrically invariant, and computationally simple: We minimise simultaneously the sum of the algebraic distances and the sum of the squared differences of the estimated unit normals and the gradients of F at the given points. The solution is found by solving a system of linear equations. Its matrix M is symmetric, positive definite, and - in the case of a piecewise polynomial approximation - sparse.

In addition, one may consider another quadratic functional, called the tension term, which measures the deviation of F from a linear function. By simultaneously minimising the tension term and approximating the points and normals, we may 'flatten' the resulting function F. This straightens the associated implicit curve. Using this we can influence the shape of the curve. Unwanted branches, loops but also desired details like cusps or sharp edges may vanish.

In the following we describe how to compute the sum of the algebraic distances, the sum of the squared differences of the estimated normals and the gradients of F, and the tension term effectively.

4.2. Preparations

Let $\mathbf{p}_i = (p_{i,x}, p_{i,y})$ $i = 1,..,N$ denote a given set points. Our aim is to find a bivariate tensor product-spline-function,

$$F(x,y) = \sum_{(j,k)\in\mathcal{A}} c_{jk} M_j(x) N_k(y) \ ,$$

whose zero contour approximates the given set of points. Let $(M_j(x))_{j=1..m}$ and $(N_k(y))_{k=1..n}$ denote the B-spline functions of degree d with respect to certain user defined knot vectors, and $(c_{jk})_{(j,k)\in\mathcal{A}}$ be the unknown real coefficients.

In order to obtain the knot vectors, we consider a bounding box of the data, and subdivide a slightly enlarged area in quadratic cells of constant size s. This subdivision induces an equidistant grid on the x and y axis. Adding additional d knots at the beginning and at the end of the grid, we obtain the knot vectors $\mathcal{X} = (\xi_j)_{j=1..m+1}$ and $\mathcal{Y} = (\eta_k)_{k=1..n+1}$.

The domain $\mathcal{D}$ does not contain all cells within this grid. Obviously it has to contain the cells containing points, which shall be denoted by $\mathcal{D}_1$. Additionally we consider some neighbouring cells, within a user defined distance from given points. The reason for this is that the resulting curve is likely to pass through such a cell, and otherwise might be cut away. The union of these neighbouring cells shall be denoted by $\mathcal{D}_2$. We have $\mathcal{D} = \mathcal{D}_1 \dot\cup \mathcal{D}_2$, where $\dot\cup$ denotes the disjoint union.

In the following we use the bijective relation between the lower left corner of the cells and the pair of indices (j,k) associated with the knots. The domain $\mathcal{D}$, its subsets $\mathcal{D}_i$, as well as single cells can be identified with certain subsets of $\mathcal{K} = \{1,..,m\} \times \{1,..,n\}$.

Due to the restriction of F on $\mathcal{D}$, we only need to take the products of basis functions $M_p(x)N_q(y)$ into account that do not vanish on $\mathcal{D}$. The indices are:

$$\mathcal{A} = \{(p,q) \in \mathcal{K} | \exists (j,k) \in \mathcal{D} \wedge j-d \le p \le j \wedge k-d \le q \le k\}$$

Note that, for some of these 'active' B-spline functions the support may not contain any given points $\mathbf{p}_i$. This fact however may only occur when neighbour cells were included in $\mathcal{D}_2$. In figure 4 these cells are marked by a cross.

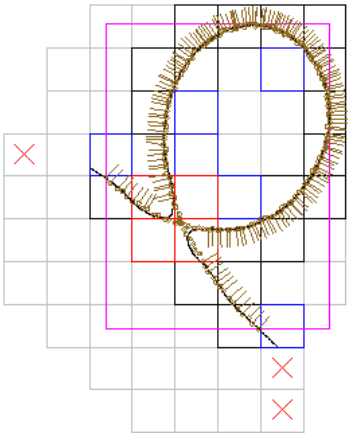

Figure 4: *Bounding box, domain $\mathcal{D}$, and active B-spline functions*

In order to simplify the notation, we enumerate $\mathcal{A}$. Thus the coefficients c_{jk} are collected in a vector $\vec{c}$ of length h with $h = |\mathcal{A}|$. Each pair of indices (j,k) is bijectively related to an index $s \in \{1,\ldots,h\}$.

4.3. Algebraic distance

The sum of the squared algebraic distance of the data points $\mathbf{p}_i$ from the implicit curve $F(x,y) = 0$ determined by the coefficients $\vec{c}$ is given by

$$\delta(\vec{c}) = \sum_{i=1}^{N} (F(\mathbf{p}_i))^2 = \sum_{i=1}^{N} \left(\sum_{(j,k)\in\mathcal{A}} M_j(x_i)N_k(y_i)\, c_{jk} \right)^2 .$$

In matrix notation this quadratic form can be written as

$$\delta(\vec{c}) = \vec{c}^{\,\top} \cdot D \cdot \vec{c}.$$

The matrix-elements

$$d_{jk,lm} = \sum_{i=1}^{N} M_j(x_i)N_k(y_i)M_l(x_i)N_m(y_i) \qquad (3)$$

are supposed to be ordered (row and column wise) in the same way as the coefficients $\mathbf{c}_{jk}$.

For any arbitrary order of the (c_{jk}) we have that D is a $h \times h$ symmetric, positive semi-definite matrix, since $d_{jk,lm} = d_{lm,jk}$ and $\delta(\vec{c}) \ge 0$. As a consequence we have to build only the upper triangular part of D. Moreover, due to the limited support of the B-spline basis functions, we can state that D is sparse, since most of the products $M_j(x)N_k(y)M_l(x)N_m(y)$ vanish identically. For a fixed pair of indices (l,m) we only need to consider the elements $d_{jk,lm}$ with

$$(j,k) \in \mathcal{A}_{l,m} = \{l-d, l+d\} \times \{m-d, m+d\} \cap \mathcal{A}.$$

In order to adapt the algorithm to problematic input data, we now introduce the following weight function:

$$\omega^{\delta}(\mathbf{p}_i) = \begin{cases} w_1^{\delta} & \text{if type}(\mathbf{p}_i) = 1, \\ w_2^{\delta} & \text{if type}(\mathbf{p}_i) = 0, \end{cases}$$

and change the summation in (3) as follows: We intersect the support of $M_j(x)N_k(y)M_l(x)N_m(y)$ with the domain $\mathcal{D}$

$$\begin{aligned} \mathcal{D}_{jk}^{lm} = {} & \{\max(j,l), \min(j+d, l+d)\} \times \\ & \{\max(k,m), \min(k+d, m+d)\} \cap \mathcal{D}, \end{aligned}$$

and then introduce a data structure, which links the cells and points. Let $\mathbf{p}_i^{pq} = (x_i^{pq}, y_i^{pq})$ denote the points in the cell with indices (p,q), and let N_{pq} be the number of them. The elements $d_{jk,lm}$ now can be written as

$$d_{jk,lm} = \sum_{(p,q)\in\mathcal{D}_{jk}^{lm}} \sum_{i=1}^{N_{pq}} \omega^{\delta}(\mathbf{p}_i^{pq}) M_j(x_i^{pq}) N_k(y_i^{pq}) M_l(x_i^{pq}) N_m(y_i^{pq}). \qquad (4)$$

This shows how to generate D in a fast and efficient way.

4.4. Approximation of the normals

The sum of the squared differences of the estimated normals $\mathbf{n}_i$ and the gradient field of F evaluated at the data points $\mathbf{p}_i$ is given by

$$\nu(\vec{c}) = \sum_{i=1}^{N} \omega^{\nu}(\mathbf{p}_i) \left(\left(F_x(\mathbf{p}_i) - n_{i,x}\right)^2 + \left(F_y(\mathbf{p}_i) - n_{i,y}\right)^2 \right).$$

where

$$\omega^{\nu}(\mathbf{p}_i) = \begin{cases} w^{\nu} & \text{if type}(\mathbf{p}_i) = 1, \\ 0 & \text{if type}(\mathbf{p}_i) = 0. \end{cases}$$

Again we may express $\nu(\vec{c})$ in matrix notation

$$\nu(\vec{c}) = \vec{c}^{\,\top} \cdot G \cdot \vec{c} \; - \; 2\vec{r}^{\,\top} \cdot \vec{c} \; + \; \sum_{i=0}^{N} \omega^{\nu}(\mathbf{p}_i) \|\mathbf{n}_i\|^2.$$

where the elements of G - using the notation from the previous section - are

$$\begin{aligned} g_{jk,lm} = {} & \sum_{(p,q)\in\mathcal{D}_{jk}^{lm}} \sum_{i=1}^{N_{pq}} \omega^{\nu}(\mathbf{p}_i^{pq}) \\ & \left(\dot{M}_j(x_i^{pq}) N_k(y_i^{pq}) \dot{M}_l(x_i^{pq}) N_m(y_i^{pq}) \right. \\ & \left. + M_j(x_i^{pq}) \dot{N}_k(y_i^{pq}) M_l(x_i^{pq}) \dot{N}_m(y_i^{pq}) \right). \end{aligned}$$

In order to obtain $\vec{r}$, we intersect $\mathcal{D}$ with the support of $M_j(x)N_k(y)$ (i.e. $\mathcal{D}_{jk} = \{j, j+d\} \times \{k, k+d\} \cap \mathcal{D}$). We arrive at

$$\vec{r}_{jk} = \sum_{(p,q)\in\mathcal{D}_{jk}} \sum_{i=1}^{N_{pq}} \omega^{\nu}(\mathbf{p}_i^{pq}) \left(\dot{M}_j(x_i^{pq}) N_k(y_i^{pq}) n_{i,x}^{pq} + M_j(x_i^{pq}) \dot{N}_k(y_i^{pq}) n_{i,y}^{pq} \right).$$

4.5. Tension

A function 'measuring' the deviation of a polynomial function F from a linear function is given by the quadratic functional

$$\tau(\vec{c}) = \iint_{\mathcal{D}} F_{xx}^2 + 2F_{xy}^2 + F_{yy}^2 \; \mathbf{d}x\, \mathbf{d}y\,.$$

As before we use the matrix notation

$$\nu(\vec{c}) = \vec{c}^{\,\top} \cdot T \cdot \vec{c}.$$

Let D_j be the interval (ξ_j, ξ_{j+1}) - respective (η_j, η_{j+1}). The elements of T are

$$t_{jk,lm} = \sum_{(p,q)\in\mathcal{D}_{jk}^{lm}} \left(\int_{\mathcal{D}_p} \ddot{M}_j(x)\ddot{M}_l(x)\,\mathbf{d}x \int_{\mathcal{D}_q} N_k(y)N_m(y)\,\mathbf{d}y + 2\int_{\mathcal{D}_p} \dot{M}_j(x)\dot{M}_l(x)\,\mathbf{d}x \int_{\mathcal{D}_q} \dot{N}_k(y)\dot{N}_m(y)\,\mathbf{d}y + \int_{\mathcal{D}_p} M_j(x)M_l(x)\,\mathbf{d}x \int_{\mathcal{D}_q} \ddot{N}_k(y)\ddot{N}_m(y)\,\mathbf{d}y \right)$$

Due to the equidistant gridsize, all the B-spline functions $M_r(x)$ and $M_s(x)$ are related by a simple linear parameter transformation. As a consequence, the above integrals with respect to x depend (for fixed degree d and gridsize s) only on the difference $|j-l|$ and on $p-\min\{j,l\}$. So, for fixed degree d, we may compute a total of $3(d+2)(d+1)/2$ different numerical values in advance, and store them in a look-up table. The integrals with respect to y yield the same values, since the grids in x- and y- direction were assumed to be equally spaced.

The effect resulting from different tension weights can be seen in figures 5 and 6.

4.6. Approximative implicitization

As already mentioned, our aim is to solve the quadratic minimisation problem

$$\delta(\vec{c}) + \nu(\vec{c}) + w^{\tau}\tau(\vec{c}) \to \min, \tag{5}$$

in order to get an approximation for the problem $\delta(\vec{c}) \to \min$.

The solution of (5) is given by

$$\underbrace{(D+G+w^{\tau}T)}_{M} \cdot \vec{c} = \vec{r}, \tag{6}$$

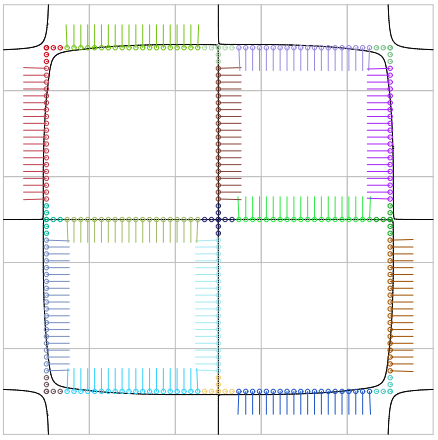

Figure 5: $w^{\tau} = 1 \cdot 10^{-6}$

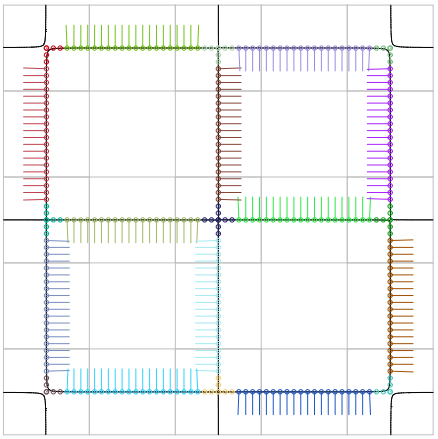

Figure 6: $w^{\tau} = 0$

where M is a symmetric positive definite matrix. As shown previously[12] (5) is a positive, strictly convex functional if the weights are all positive. Consequently (6) has a unique solution.

The normals term $\nu(\vec{c})$ has been introduced due to the need of a 'normalisation' of the solution space of $\delta(\vec{c}) \to \min$. The weight w^{ν} is chosen as low as possible.

The term $\tau(\vec{c})$ is used to influence the topology of the resulting curve. It can be omitted if $\mathcal{D}_2$ is empty. Otherwise neighbouring cells are considered for the domain $\mathcal{D}$. In this case, it may occur that certain coefficients $\mathbf{c}_{jk}$ are influenced only by the tension term. Hence M would be singular if $\tau(\vec{c})$ was omitted.

4.7. Fallback strategy for normal estimation

Clearly the process of normal estimation, segment building and orientation propagation depends on the sampling density

of the input data, the amount of noise and the choice of the parameters k, l and C. Currently we choose C rather small, consequently criterion (2) is restrictive. This results in more points and hence larger segments of type 0. So the process of orientation transfer is stabilised. Unintentional segments of type 0 may occur, for example in regions of high curvature. This is no real drawback, since we only use the estimated normals as additional information.

We do not have to obtain estimates for all of the points. A secure global orientation is of greater importance. This is exploited in a simple fall-back strategy:

Whenever we detect an error in the step of orientation propagation, we consider only the normals of the largest segment $\tilde{S}$ of type 1. For all other points the type is set to 0. We then compute an initial approximative implicitization for this input data. Numerical results show that using only a part of the normals still leads to an acceptable approximation.

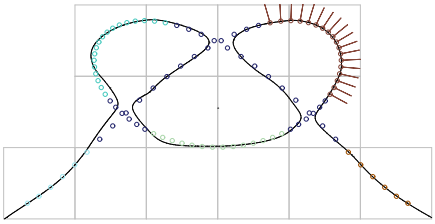

Figure 7: *Initial step*

However symmetries in the input data are not automatically inherited by the resulting curve. This is resolved comparing the estimated normal vectors with the gradients of the initial implicitization in the given data points. Using the same orientation for this vectors, we get a reliable unit vector field, which is used to compute the final algebraic fit.

Figure 8 is based on the same data as figure 7. Yet the weight of the tension term is chosen intentionally to low. The resulting curve approximates the data, but does not have the correct topology, which is needed for the adjustment of the estimated normals. The updated normals field is shown in figure 9. Most of the normals of the top left segment have a wrong orientation. Based on this data, we show the usefulness of the iteration process presented in the following section. After ten additional iteration steps, we obtain the nearly symmetric approximation, shown in figure 10.

4.8. Iteration

Clearly, the result of the approximation process is substantially influenced by the reliability of the estimated normals. In order to get better estimates, we iterate the approximation step. After each approximation step we compute the gradients of the approximating function F in the given data points, and use them as new estimates.

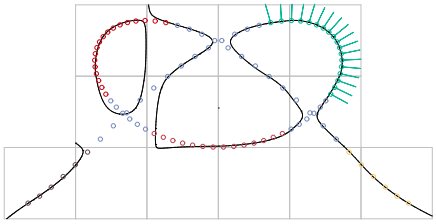

Figure 8: *Initial step for a low weight* w^τ

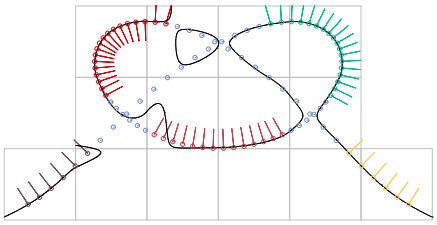

Figure 9: *Adjusted normals*

In each step we minimise simultaneously the sum of the algebraic distances as well as the difference of the estimated normal vectors and the gradients. Consequently, after each iteration step the gradients should give a better estimate of the normals.

In order to avoid contraction of F towards the zero function (with vanishing gradients everywhere), we have to scale the computed gradients after each step, before we can use them as new estimates $\mathbf{n}_i$. We scale them such that the sum of the squared lengths of the $\mathbf{n}_i$ equals N.

$$\mathbf{n}_i = \sqrt{\frac{N}{\sum_{j=0}^{N} \|\nabla F(\mathbf{p}_j)\|^2}}\,\nabla F(\mathbf{p}_i)$$

Note that the matrix M of the linear system does not depend on the estimated normals. Only the right-hand side $\vec{r}$ has to be updated after each adaptation of the estimates. Computing a Cholesky decomposition of M once, in each iteration step we only have to apply back and forward substitution to obtain the algebraic fit. The right-hand side $\vec{r}$ depends linearly on the estimates $\vec{n}_i$, which depend again linearly on the

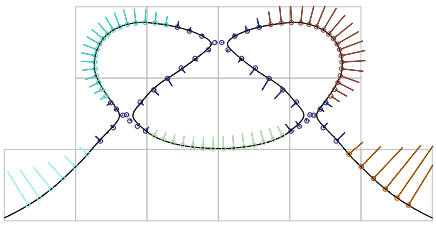

Figure 10: *After 10 iteration steps*

previous computed coefficients of F. Summing up, the computational costs of the iteration process are very low.

After the initial approximation step we get estimates for the points that failed criterion (2). Consequently the distinction of points that have / do not have an estimated normal can be considered as obsolete. The type of all points can be set to 1. However, if we do this, we need to rebuild the matrix M, since the matrices D and G depend on the type of the points. Similarly, when applying the fallback strategy, M has to be rebuilt, in order to avoid asymmetries introduced by the choice of the initial segment. The same is true for changes of the weights during the iteration. Consequently a 'reweight procedure' as proposed in [12, 16] seems to be rather disadvantageous.

4.9. Choice of degree

An important question concerning the fitting process is the choice of the degree d. From the following lemma we get a lower bound on d. Note, that using a tensor product representation favours the x- and y-directions.

Lemma 1. *The zero contour c of a tensor product surface $\sum_{j,k=0}^{d} c_{j,k} x^j y^k$ of bi-degree d may have a k-fold point P, with general k tangent directions, only if $d \geq k$.*

Proof Let $P(a,b)$ be a k-fold point of c. The partial derivatives of F vanish in P up to order $k-1$, and there exists at least one partial derivative of order k that is not zero. The k tangents of c in P are the lines $(a+\lambda t, b+\mu t)$, with:

$$\sum_{j=0}^{k} \binom{k}{j} \frac{\partial^k F}{\partial x^{k-j} \partial y^j}\Big|_{(a,b)} \lambda^{k-j} \mu^j = 0. \qquad (7)$$

The y-axis (x-axis) parallel is a tangent exactly if $\lambda = 0$ ($\mu = 0$) fulfils (7). This is equivalent to $\frac{\partial^k F}{\partial y^k}|_{(a,b)} = 0$ ($\frac{\partial^k F}{\partial x^k}|_{(a,b)} = 0$). If $d < k$ this last condition is clearly always fulfilled. If $d \geq k$, however, the partial derivatives depend on the coefficients c_{jk}, and hence may but need not to vanish. □

Given scattered data, a lower bound for the degree can be found as follows: A segment of type 0 having m neighbour segments is assumed to represent a point of multiplicity $m/2$.

As an example figures 11 and 12 show the degree 3 and 4 polynomial approximation of four (non-axes-parallel) lines coinciding in one point. These four lines can be interpreted as one implicit curve which has an 4-fold point. In accordance with the lemma, the degree 3 approximation clearly fails to approximate the given data. Instead it resembles four pairwise parallel lines, that can be interpreted as implicit curve having four double points.

5. Bézier approximation

In order to search for an approximative implicitization in the sense of the last section, now using a polynomial basis, only few adaptations have to be made. The domain is now

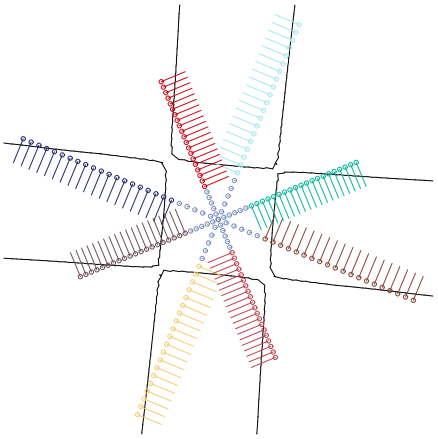

Figure 11: *Degree* 3 *Bézier-approximation*

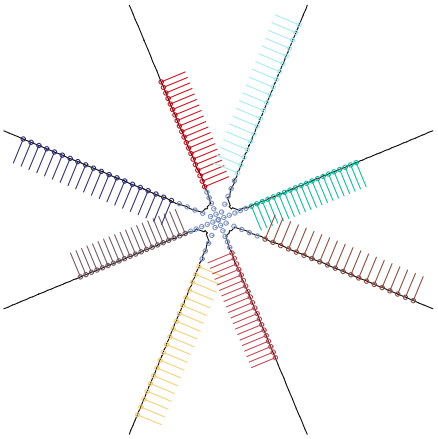

Figure 12: *Degree* 4 *Bézier-approximation*

a slightly enlarged bounding box $D = (a,b) \times (c,d)$ of the input data. Due to the expected numeric stability we use the Bernstein polynomials of degree d

$$M_j(x) = \binom{d}{j}(1-u)^{n-j}u^j \quad \text{with} \quad u = \frac{b-x}{b-a},$$
$$N_k(y) = \binom{d}{k}(1-v)^{n-k}v^j \quad \text{with} \quad v = \frac{d-y}{d-c},$$

with $j,k = 0,\ldots,d$ as basis functions. The support of the products $M_j(x)N_k(y)$ is no longer restricted. As a direct consequence the set of 'active products' is $\mathcal{A} = \{0,\ldots,d\} \times \{0,\ldots,d\}$. Furthermore there are no vanishing products $M_j(x_i)N_k(y_i)M_l(x_i)N_m(y_i)$ we can omit (cp. (3) and (4)). Consequently the resulting linear system is no longer sparse (but still symmetric).

Figures 4 and 13 show a comparison of the behaviour of the implicit curves for the same input data. In order to make the differences visible, we used non-optimised weights. The

number of coefficients is approximatively the same: In figure 4 we have 67 B-spline basis functions of degree 2, and in the case of figure 13 we considered the 64 products of Bernstein polynomials of degree 7. The 'local control' property of splines leads to a better approximation.

Each coefficient of D, G, and T is a sum of (integrals of) products of the basis functions and their derivatives. Due to the limited support of the B-spline functions the number of evaluations of such products can be reduced. As a consequence the generation of the linear system is noticeably faster for splines.

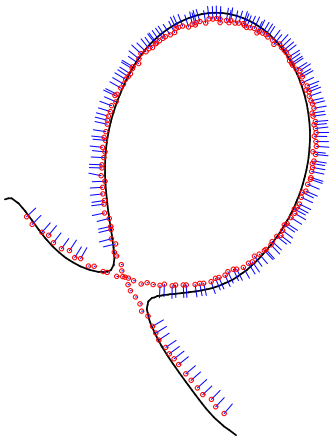

Figure 13: *Bézier-approximation*

6. Conclusion

We have described a method for fitting implicitly defined (piecewise) curves to planar scattered data, which may contain problematic regions (e.g. singular points). The algorithm is based on the simultaneous approximation of the given data and estimated normal vectors, which are generated in a preprocessing step.

In order to assure a correct global orientation of these estimates, we consider a simple criterion rejecting unreliable points, create a natural segmentation of the data and finally use methods for global orientation propagation.

Using this estimates as additional information for the task of curve fitting, we get a geometrically invariant method for curve reconstruction. It is computationally simple, as we only need to solve a linear system for the coefficients of the unknown representation.

Based on the segmentation and the approximation step, we have introduced a simple fallback strategy, in case the orientation propagation step fails. Furthermore we have shown how the segmentation can be used to obtain a lower bound for the degree of the implicit curve.

Finally we described an iteration of the process. Using the gradients of the resulting function as new, more reliable normal estimates, we may improve the approximation. In each step we only have to rebuild the right-hand side of the linear system.

Future research will address convergence properties of the iteration process, methods of reliable normal estimation and orientation propagation for surfaces, segment classification (object recognition), and possible combination with exact methods. First results for surfaces (Figure 14), relying on the fallback strategy, have already demonstrated the applicability of our approach.

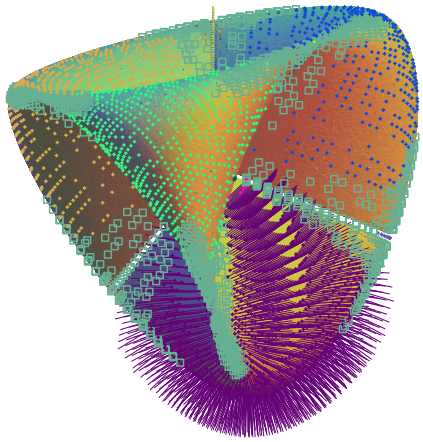

Figure 14: *Degree 2 Bézier approximation of Steiner's 'Roman surface', relying on one segment of points with estimated normals*

Acknowledgements

This research is funded by the European Commission through project IST-2002-35512 - GAIA II , entitled 'Intersection algorithms for geometry based IT-applications using approximate algebraic methods'.

References

1. C.L. Bajaj, I. Ihm, J. Warren, "Higher-order interpolation and least-squares approximation using implicit algebraic surfaces". *ACM Transactions on Graphics* **12**, 327-347, 1993
2. F.Bernardini, C.L. Bajaj, J. Chen, D.R. Schikore, "Automatic reconstruction of 3D CAD models from digital scans". *Int. J. Comp. Geom. Appl.* **9**, 327-370, 1999
3. U. Clarenz, M. Rumpf, and A. Telea, "Robust Feature Detection and Local Classification for Surfaces based on Moment Analysis". *submitted*
4. J.H.Chuang, C.M. Hoffmann, "On local implicit approximation and it's applications". *ACM Trans. Graphics* **8**, 4, 298-324, 1989

5. D. Cox, J. Little, D. O'Shea T.W. Sederberg, F. Chen, Ideals, Varieties and Algorithms, Springer Verlag, New York, 1992 & 1997.

6. D. Cox, J. Little, D. O'Shea T.W. Sederberg, F. Chen, Using Algebraic Geometry, Springer Verlag, New York, 1998.

7. D. Cox, R. Goldman, M. Zhang, "On the validity of Implicitization by Moving Quadrics for Rational Surfaces with No Base Points", *J. Symbolic Computation* **11**, 1999.

8. T. Dokken, "Approximate Implicitization". Mathematical methods in CAGD: Oslo 2000, Tom Lyche and Larry L. Schumaker (eds.) pp. 1-25, 2001.

9. M. Gopi, S. Krishnan, C.T. Silva, "Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation". *Computer Graphics Forum (Eurographics 2000)* **19**(3), 2000.

10. C. M. Hoffmann, "Implicit Curves and Surfaces in CAGD", *Comp. Graphics and Applics.* **13**, 79-88, 1993.

11. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Surface reconstruction from unorganized points". *Computer Graphics (SIGGRAPH '92 Proceedings)*, **26**(2):71–78, July 1992.

12. Jüttler, B. " Least-square fitting of algebraic spline curves via normal vector estimation." The Mathematics of Surfaces IX, Roberto Cipolla and Ralph Martin (eds.), Springer London, 263-280, 2000.

13. B. Jüttler, A. Felis, "Least-square fitting of algebraic spline surfaces". *Advances in Computational Mathematics* **17** (1-2): 135-152, 2002.

14. V. Pratt, "Direct least-squares fitting of algebraic surfaces". *ACM Computer Graphics* **21**, (Siggraph'87), 145-152, 1987.

15. T.W. Sederberg, F. Chen, "Implicitization using moving curves and surfaces". *Computer Graphics* Proceedings, Annual Conference Series, **29**:301–308, 1995.

16. R. Taubin, "Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation". *IEEE Trans. Pattern Analysis and Machine Intelligence* **13**, 1115-1138, 1991