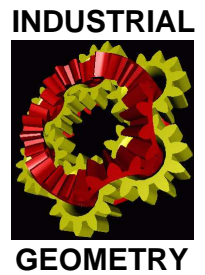


Forschungsschwerpunkt S92

Industrial Geometry

<http://www.ig.jku.at>



FSP Report No. 2

Hybrid Curve Fitting

M. Aigner, B. Jüttler

December 2005

FWF

Der Wissenschaftsfonds.



Hybrid Curve Fitting

Martin Aigner and Bert Jüttler

Institute of Applied Geometry
Johannes Kepler University, Linz, Austria
www.ag.jku.at

Abstract

We consider a parameterized family of closed planar curves and introduce an evolution process for identifying a member of the family that approximates a given unorganized point cloud $\{\mathbf{p}_i\}_{i=1,\dots,N}$. The evolution is driven by the normal velocities at the closest (or foot) points (\mathbf{f}_i) to the data points, which are found by approximating the corresponding difference vectors $\mathbf{p}_i - \mathbf{f}_i$ in the least-squares sense. In the particular case of parametrically defined curves, this process is shown to be equivalent to normal (or tangent) distance minimization, see [3, 19]. Moreover, it can be generalized to very general representations of curves. These include hybrid curves, which are a collection of parametrically and implicitly defined curve segments, pieced together with certain degrees of geometric continuity.

1 Introduction

Fitting a parameterized curve to a given set of unorganized points is an important problem in fields such as geometric modeling and computer vision. Due to the influence of the parameterization, this leads to a non-linear optimization problem. Different approaches for dealing with the effects of this non-linearity have been developed, such as ‘parameter correction’ or the use of quasi-Newton methods [2, 6, 11, 12, 13, 15, 17, 19]. Clearly the choice of a good initial solution is of outmost importance for the success of the optimization. Geometrically motivated optimization strategies [13, 14, 15, 19], where the initial solution is replaced by an initial curve and the formulation of the problem uses some geometric insights, may lead to more robust techniques.

Due to the iterative nature of the techniques for non-linear optimization, it is natural to view the intermediate results as a time-dependent curve which tries to adapt itself to the target shape defined by the unorganized point data [14, 19]. This is related to the concept of ‘active curves’ used for image segmentation in Computer Vision [18], and to so-called Level Set methods [9, 10], where shapes are defined by time-dependent discretizations of (approximations to) the signed distance function.

This paper is organized as follows. First we introduce a framework for fitting a member of a given family of planar curves to unorganized point data, by defining an evolution process which generates a time-dependent curve. For instance, the family of planar curves may consist of “hybrid objects” obtained by combining simple shapes (such as circular arcs) with free-form curves, see Figure 1 for a first example.

In the case of parametric curves, the evolution is equivalent to the method of normal (or tangent) distance minimization [3, 19], as shown in Section 3. Section 4 discusses the generalization to other classes of curves. Finally we conclude the paper.

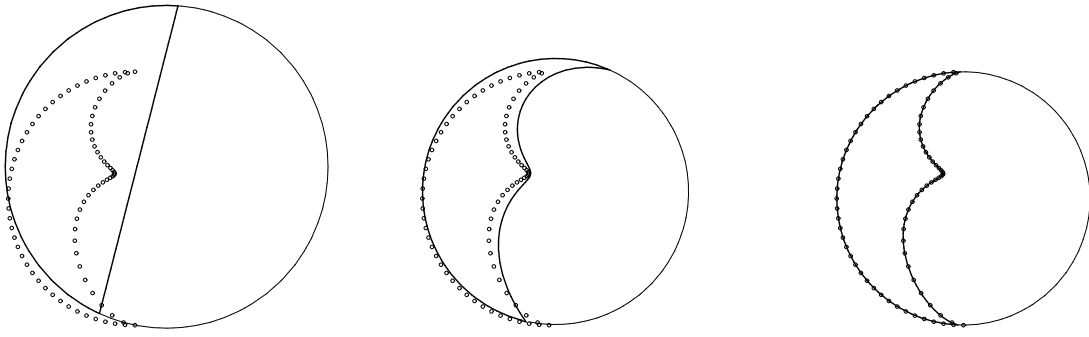


Figure 1: Fitting a hybrid object (consisting of a circular arc and a B-spline curve) to a point cloud. Left: initial configuration. Center and right: after 5 and 20 iterations. The figure shows the entire circle, but only the bold segment belongs to the model.

2 The framework

We introduce the abstract notion of a parameterized family of closed planar curves and describe the evolution of such a family, via least-squares approximation of normal velocities.

2.1 Parameterized families of closed curves

We consider a parameterized *family of closed planar curves* $(\mathbf{s}, u) \mapsto \mathbf{c}_{\mathbf{s}}(u)$. It depends on two types of parameters: a single *curve parameter* u and a *vector of shape parameters* \mathbf{s} .

For each curve of the family, the *curve parameter* u varies within the parameter domain $I = \mathbb{S}^1$, which can be identified with an interval $\hat{I} = [a, b]$ with identified boundaries. The vector $\mathbf{s} = (s_1, \dots, s_n)$ contains a system of *shape parameters* which specify the geometry of the curve. The shape parameters are allowed to vary within a certain feasible set $\Omega \subset \mathbb{R}^n$.

We assume that $\mathbf{c}_{\mathbf{s}}(u)$ depends continuously on \mathbf{s} and u , and that it is differentiable for all $(\mathbf{s}, u) \in \Omega \times \mathbb{S}^1$, except for finitely many values $W = \{w_1, \dots, w_k\}$ of the curve parameter u , which will be called the *vertex parameters*. Still, for these values of u , it is assumed to be differentiable with respect to the shape parameters \mathbf{s} . In addition, it is assumed that

$$\mathbf{c}'_{\mathbf{s}}(u) = \frac{\partial}{\partial u} \mathbf{c}_{\mathbf{s}}(u) \neq (0, 0) \quad (1)$$

holds for all $(\mathbf{s}, u) \in \Omega \times I_0$, where $I_0 = I \setminus W$. Consequently, each curve $\mathbf{c}_{\mathbf{s}}(u)$ of the family has a well-defined tangent at all points, except for the vertex parameters¹ $\mathbf{c}_{\mathbf{s}}(w_j)$, $j = 1, \dots, k$.

Example 1 Consider the family of closed planar B-spline curve of degree d with a certain periodic knot vector, where the knots have at most multiplicity $d - 1$. The knot vector is assumed to be fixed, while the control points may vary. In this situation, the shape parameters are the components of all control points, which can be collected in a single vector \mathbf{s} of dimension $2m + 2$, where $m + 1$ is the number of control points. The set of vertex parameters $W = \{w_1, \dots, w_k\}$ consists of all knots with multiplicity $d - 1$. The set Ω contains all control points which lead to curves that are regular everywhere, except at

¹These vertices should not be confused with points of stationary curvature.

the vertices. While an exact description of the set Ω is generally hard to obtain, simple conditions for subsets can be derived easily. See also section 3.

Example 2 For a certain domain $D \subset \mathbb{R}^2$ we consider all algebraic curves (i.e., the zero level sets of polynomials) of degree d which possess exactly one closed loop within D without singular points, and no other points. The shape parameters are the coefficients of the defining polynomial, and the set Ω contains all coefficients that correspond to curves satisfying these criteria. Again, an exact description of Ω is hard to obtain, but simple descriptions for subsets are available. In this example, the curve parameterizations cannot be constructed explicitly. Nevertheless, they exist, since one may choose arc length parameterizations scaled to a suitable common parameter domain². As we shall see later, an explicit description is not needed for the fitting. The set of vertex parameters is empty.

Remark 3 The feasible set Ω will not be analyzed in this paper; we will simply assume that the distribution of the data guarantees the regularity of the resulting curve. As for the existing methods of curve fitting, one may use variational techniques [4], whenever this assumption is violated.

2.2 Fitting by evolution

Consider a given unorganized set of data points $\{\mathbf{p}_j\}_{j=1..N}$ and a given parameterized family of closed planar curves with shape parameters $\mathbf{s} = (s_1, \dots, s_n)$, where $n \ll N$. We define an evolution process in order to identify a curve among the family which approximates these data.

Now, the shape parameters $\mathbf{s} = \mathbf{s}(t) = (s_1(t), \dots, s_n(t))$ depend smoothly on an *evolution parameter* t , which be identified with the time. This leads to a smoothly varying one-parameter family of curves. For $t \rightarrow \infty$, the evolution produces a stationary value $\mathbf{s}_0 = \lim_{t \rightarrow \infty} \mathbf{s}(t)$ which defines the approximating curve.

At a regular point $\mathbf{c}_{\mathbf{s}(t)}(u_0)$ (i.e., $u_0 \notin W$), the normal velocity of the curve equals

$$v(u_0) = \sum_{i=1}^n \frac{\partial \mathbf{c}_{\mathbf{s}}(u_0)}{\partial s_i} \dot{s}_i(t) \Big|_{\mathbf{s}=\mathbf{s}(t)} \cdot \vec{\mathbf{n}}_{\mathbf{s}(t)}(u_0), \quad (2)$$

where $\vec{\mathbf{n}}_{\mathbf{s}}(u_0)$ is the normal vector of the curve at $u = u_0$, and the dot denotes the derivative with respect to t . Note that the normal velocity depends *linearly* on the time derivatives of the shape parameters.

The *evolution* is governed by a system of ordinary differential equations of the form $\dot{\mathbf{s}} = F(\mathbf{s})$. In order to derive this system, we consider – for a certain time t and associated shape parameters $\mathbf{s} = \mathbf{s}(t)$ – the closest point(s), or foot point(s)³, $\mathbf{f}_j = \mathbf{c}_{\mathbf{s}}(u_j)$ on the curve which can be associated with each data point \mathbf{p}_j . Clearly, the parameter u_j of this point depends on the shape parameters \mathbf{s} . In order to move the curve closer to the data, the closest points are expected to travel with the normal velocity

$$d_j := (\mathbf{p}_j - \mathbf{f}_j) \cdot \vec{\mathbf{n}}_{\mathbf{s}}(u_j) \quad (3)$$

²Strictly speaking, one would need to define a suitable starting point for these arc length parameterizations, e.g., the point where one of the coordinates attains its minimum value. In order to make sure that this choice is feasible, one may consider only algebraic curves where this point is unique.

³In each time step, robust and efficient techniques for computing the closest points (foot points) of the data on the current curve are needed, cf. [1, 7, 16]

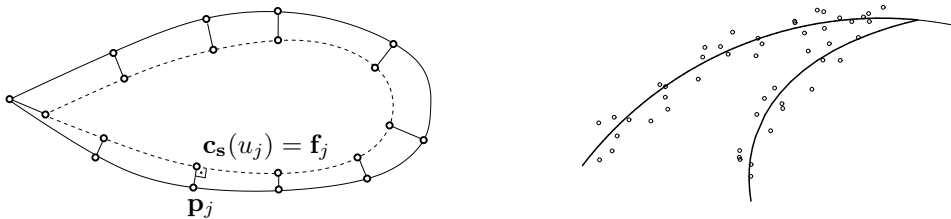


Figure 2: Left: Fitting a curve to a point cloud by evolution. Dashed/solid line is before/after displacement. Right: Difficulties with sharp corners (see conclusion)

Clearly, this normal velocity is defined only for regular points, $u_j \notin W$. If one of the closest points \mathbf{f}_j falls into a vertex, $u_j \in W$, then its velocity

$$\vec{v}_{\text{vertex}}(u_j) = \sum_{i=1}^n \frac{\partial \mathbf{c}_{\mathbf{s}}(u_j)}{\partial s_i} \dot{s}_i(t) \Big|_{\mathbf{s}=\mathbf{s}(t)}, \quad (4)$$

which depends again linearly on the \dot{s}_i , is expected to be equal to the difference $\mathbf{p}_j - \mathbf{f}_j$.

Generally, both conditions cannot be satisfied exactly for all points, since the points depend on the shape parameters \mathbf{s} , and $n \ll N$. Instead, we choose the time derivatives $\dot{\mathbf{s}}$ of the shape parameters such that both conditions are satisfied in the least-squares sense,

$$\dot{\mathbf{s}} = \arg \min_{\dot{\mathbf{s}}} = \sum_{\substack{j=1, \dots, N \\ u_j \notin W}} (v(u_j) - d_j)^2 + \sum_{\substack{j=1, \dots, N \\ u_j \in W}} \|\vec{v}_{\text{vertex}}(u_j) - (\mathbf{p}_j - \mathbf{f}_j)\|^2 + R. \quad (5)$$

The regularization term R is added in order to guarantee a unique solution. For instance, one may choose a Levenberg–Marquardt strategy, $R = \omega \|\dot{\mathbf{s}}\|^2$, with a small positive weight $\omega > 0$, cf. [8]. Due to the linear influence of the time derivatives \dot{s}_i in (2) and (4), Eq. (5) leads to a system of *linear* equations

$$M(\mathbf{s}) \dot{\mathbf{s}} = \mathbf{r}(\mathbf{s}) \quad \text{or, equivalently,} \quad \dot{\mathbf{s}} = M(\mathbf{s})^{-1} \mathbf{r}(\mathbf{s}). \quad (6)$$

with a symmetric positive definite matrix M , which defines the *evolution* of the curve. In order to trace the evolving one parameter–family of curves, we use explicit Euler steps⁴ for numerically solving (6). In each time step, instead of computing the inverse matrix M^{-1} , we find the values of the time derivatives $\dot{\mathbf{s}}$ by solving the linear system.

Remark 4 The evolution by (normal) velocities defined by the closest points works reasonably well if the current curve is already relatively close to the data. Otherwise, some regions of the curve (without closest points) may simply be ignored by the evolution, since the Levenberg–Marquardt regularization tends to ‘freeze’ the previous position.

In order to address this problem, one may use a two–step evolution process (see [20]), as follows. As a preprocessing, one generates the unsigned distance field U of the given points. For instance, this can be done efficiently by using graphics hardware [5]. In the first step of the evolution, one defines an equally spaced set of ‘sensor points’ along the

⁴Instead, one may use (e.g.) higher order Runge–Kutta methods for tracing the solutions of (6). However, since the stationary state reached in the limit $t \rightarrow \infty$ is more interesting than the path that leads to this solution, the simple explicit Euler method suffices.

curve and defines the normal velocities (resp. velocities in the case of vertices) at these points based on the unsigned distance field, $v = \nabla U \cdot \vec{\mathbf{n}}$. This evolution drives the curve close to the data. The use of ‘sensor’ points corresponds to defining the force by numerical integration along the curve. In the second step, one may replace the sensor points by closest points, and use the previously described (normal) velocities.

Example 5 (Continuation of Example 2) In the case of algebraic curves,

$$\mathcal{C}_{\mathbf{s}(t)} := \{ \mathbf{x} \in D \mid q_{\mathbf{s}(t)}(\mathbf{x}) = 0 \}, \quad (7)$$

where the shape parameters \mathbf{s} are the time –dependent coefficients of the time–dependent polynomial $q_{\mathbf{s}(t)}$, the normal velocity at a point of the zero contour equals $-\dot{q}_{\mathbf{s}(t)} / \|\nabla q_{\mathbf{s}(t)}\|$, where $\vec{\mathbf{n}}_{\mathbf{s}} = \nabla q_{\mathbf{s}(t)} / \|\nabla q_{\mathbf{s}(t)}\|$. Problem (5) leads to

$$\dot{\mathbf{s}} = \arg \min_{\dot{\mathbf{s}}} = \sum_{j=1, \dots, N} \left(-\frac{\dot{q}_{\mathbf{s}(t)}(\mathbf{f}_j)}{\|\nabla q_{\mathbf{s}(t)}(\mathbf{f}_j)\|} - d_j \right)^2 + R. \quad (8)$$

This least–squares problem leads to a linear system for the time derivatives of the shape parameters. The parameterization of the curve is not needed; it suffices to know the closest points (foot points) \mathbf{f}_j which are associated with the data. Note that – especially in the case of polynomials of higher degree – the regularization should include terms that guarantee that the scalar field defined by $q_{\mathbf{s}(t)}$ approximates the signed distance field of the curve, at least within the neighborhood of the curve. Otherwise, unwanted branches and singular points are likely to occur. This simultaneously serves to normalize the coefficients of the curve. See [20] for details.

3 Parametric curves

In this section we analyze the case of a family of closed B-spline curves. It is shown that the evolution process described in Section 2.2 indeed produces a curve approximating the data, and that the explicit Euler is closely related to the technique of normal (tangent) distance minimization [3, 19].

3.1 Fitting by evolution

We consider a time–dependent family of closed planar B-spline curves of degree d ,

$$\mathbf{c}_{\mathbf{s}(t)}(u) = \sum_{i=0}^m \phi_i(u) \mathbf{b}_i(t), \quad (9)$$

see Example 1. The B-splines ϕ_i are defined with respect to a periodic knot vector, and the control points $\mathbf{b}_i = \mathbf{b}_i(t)$ which are represented as column vectors in \mathbb{R}^2 depend on the time t . For the sake of simplicity we exclude the case of knots with multiplicity $d - 1$. The vector of shape parameters contains the components of all control points, $\mathbf{s} = (b_{0,x}, b_{0,y}, b_{1,x}, b_{1,y}, \dots, b_{m,x}, b_{m,y})$. The velocity of a fixed curve point $\mathbf{c}_{\mathbf{s}(t)}(u_0)$ is the time derivative of (9), which will be indicated by a dot.

We apply the framework of Section 2.2 to this situation. Problem (5) leads to

$$\dot{\mathbf{s}} = \arg \min_{\dot{\mathbf{s}}} \sum_{j=1}^N \left[\underbrace{\left(\sum_{i=0}^m \phi_i(u_j) \dot{\mathbf{b}}_i(t) - \mathbf{p}_j + \mathbf{f}_j \right)^\top}_{=: F} \vec{\mathbf{n}}_{\mathbf{s}(t)}(u_j) \right]^2, \quad (10)$$

where $\mathbf{f}_j = \mathbf{c}_s(t)(u_j)$ is the closest point of \mathbf{p}_j on the curve, and $\vec{\mathbf{n}}_s(u)$ is the unit normal vector at $\mathbf{c}_s(u)$, which is obtained by a counterclockwise rotation of 90° from the unit tangent vector.⁵

Differentiating F with respect to the time derivatives of the control points leads to the system of $2m + 2$ linear equations

$$\sum_{j=1}^N \sum_{i=0}^m \phi_i(u_j) \phi_k(u_j) \vec{\mathbf{n}}_j \vec{\mathbf{n}}_j^\top \dot{\mathbf{b}}_i = \sum_{j=1}^N (\mathbf{p}_j - \mathbf{f}_j) \phi_k(u_j), \quad k = 0, \dots, m, \quad (11)$$

where $\vec{\mathbf{n}}_j = \vec{\mathbf{n}}_s(u_j)$. This can be rewritten in the matrix form

$$\sum_{j=1}^N \left[\phi_i(u_j) \phi_k(u_j) \vec{\mathbf{n}}_j \vec{\mathbf{n}}_j^\top \right]_{i,k=0..m} \left[\dot{\mathbf{b}}_i \right]_{i=0..m} = \left[\sum_{j=1}^N (\mathbf{p}_j - \mathbf{f}_j) \phi_k(u_j) \right]_{k=0..m} \quad (12)$$

By solving this system we compute the $\dot{\mathbf{b}}_i$ and update the \mathbf{b}_i by an explicit Euler step.

3.2 Stationary points

We show how the evolution of parametric curves defined by (10) is related to the usual fitting procedures for least-squares curve fitting. These techniques solve a problem of the form

$$\arg \min_{\mathbf{b}, \mathbf{u}} \sum_{j=1}^N \left\| \sum_{i=1}^m \phi_i(u_j^*) \mathbf{b}_i - \mathbf{p}_j \right\|^2, \quad (13)$$

where both the control points $\mathbf{b} = (\mathbf{b}_0, \dots, \mathbf{b}_m)$ and the parameter values $\mathbf{u} = (u_0^*, \dots, u_N^*)$ associated with the given data $\{\mathbf{p}_j\}_{j=0, \dots, N}$ are subject to the optimization.

In some cases (e.g. if the number of data points is too small, or if they lie in some degenerated configuration) it is not possible to get a unique solution for the curve evolution, without using a regularization.

Definition 6 For a given closed B-spline curve $\mathbf{c}_s(u)$, consider a sequence $(u_j^*)_{j=1..N}$ of parameter values and the corresponding unit normals $\vec{\mathbf{n}}_j = \vec{\mathbf{n}}_s(u_j)$. The parameters $\{u_j^*\}_{j=1..N}$ are said to be **regular** if the $(2m + 1) \times N$ matrix $[\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N]$ which is defined by the $2(m + 1)$ -dimensional vectors $\mathbf{X}_j := [\phi_i(u_j^*)]_{i=0..m} \odot \vec{\mathbf{n}}_j$ has maximal rank, where \odot denotes the Kronecker product.⁶

Proposition 7 In a regular situation, any solution of the minimization problem (13) is a stationary point of the differential equation (12).

Proof Let the u_j^* and \mathbf{b}_i^* be such that they minimize (13). By differentiation it follows that

$$\frac{\partial}{\partial \mathbf{b}_k} \left(\sum_{j=1}^N \left\| \sum_{i=0}^m \phi_i(u_j^*) \mathbf{b}_i - \mathbf{p}_j \right\|^2 \right) \Big|_{\mathbf{b}_i = \mathbf{b}_i^*} = 2 \sum_{j=1}^N \left(\sum_{i=0}^m \phi_i(u_j^*) \mathbf{b}_i^* - \mathbf{p}_j \right) \phi_k(u_j^*) = 0. \quad (14)$$

⁵In order to facilitate the theoretical analysis, the regularization term has been omitted. However, this may cause degeneracies in practice, as described later.

⁶In the case of vectors $(a_1 \dots a_n)^T \odot (b_1 \dots b_m)^T = (a_1 b_1, \dots, a_1 b_m, a_2 b_1, \dots, a_2 b_m, \dots, a_n b_1, \dots, a_n b_m)^T$

Consequently, since the u_j^* are also the parameters associated with the foot points of the data \mathbf{p}_j , the right-hand side of (12) vanishes. On the other hand, the matrix of the linear system (12) is the sum of N rank 1 matrices of the form $\mathbf{X}_j \otimes \mathbf{X}_j^\top$, where \otimes denotes the dyadic product and $\mathbf{X}_j := [\phi_i(u_j^*) \vec{\mathbf{n}}_j^\top]_{i=0..n}^\top$. In a regular case, the $2(n+1)$ matrices in the sum (12) are linearly independent and therefore the matrix is regular. Consequently, the homogenous system (12) has only the trivial solution. \blacksquare

3.3 Euler update vs. normal (tangent) distance minimization

In each time step, we compute the time derivatives $\dot{\mathbf{s}}$ and use them to generate an update of the control points by an explicit Euler step with some stepsize h ,

$$\mathbf{b}_i(t+h) = \mathbf{b}_i(t) + h\dot{\mathbf{b}}_i(t), \quad (15)$$

where the derivatives $\dot{\mathbf{b}}_i$ are found by solving (10). We compare this Euler update with the quasi-Newton technique of *normal* (or *tangent*) distance minimization [3, 19], which has been proposed for solving problem (13), see also [13]. This technique iteratively updates the control points,

$$\mathbf{b}_i \rightarrow \mathbf{b}_i + \Delta\mathbf{b}_i \quad (16)$$

where the $\Delta\mathbf{b}_i$ are found by solving

$$[\Delta\mathbf{b}_i]_{i=0,\dots,m} = \arg \min_{\Delta\mathbf{b}_i} \sum_{j=1}^N \left\| \left(\sum_{i=0}^m \phi_i(u_j) (\mathbf{b}_i + \Delta\mathbf{b}_i) - \mathbf{p}_j \right)^\top \vec{\mathbf{n}}_s(u_j) \right\|^2. \quad (17)$$

After each step, new parameters u_j are found by closest point computation.

Proposition 8 *If $h = 1$, then (15) and (16) are equivalent.*

Proof. Since $h = 1$, $\Delta\mathbf{b}_i = \dot{\mathbf{b}}_i$. Due to $\mathbf{f}_j = \mathbf{c}_s(u_j)$,

$$\left(\sum_{i=0}^m \phi_i(u_j) (\mathbf{b}_i + \Delta\mathbf{b}_i) - \mathbf{p}_j \right)^\top \vec{\mathbf{n}}_s(u_j) = \left(\sum_{i=0}^m \phi_i(u_j) \dot{\mathbf{b}}_i - (\mathbf{p}_j - \mathbf{f}_j) \right)^\top \vec{\mathbf{n}}_s(u_j),$$

the update (16) with $\Delta\mathbf{b}_i$ found from (17) and the update (15) with $\dot{\mathbf{b}}_i$ found from (10) give equivalent results. \blacksquare

4 Evolution of hybrid objects

The framework described in Section 2 can be applied not only to implicitly defined curves (see Example 5) or parametric ones (previous section), but it is also useful for *hybrid objects*. These families of curves may be designed by using additional a-priori knowledge about the target shape defined by the data, whenever such knowledge is available.

As a representative (though artificial) example, we consider a closed planar curve which is composed of two cubic Bézier curves (blue and red) and two implicitly defined quadratic curves (i.e., two conics, shown in green), see Figure 3, left. The quadratic bivariate polynomials which define the conics are described by their Bernstein-Bézier representations with respect to two basis triangles. The coefficients at the top vertices are fixed to be 1, and the coefficients at the lower four vertices are chosen to be 0, in order to ensure that the conics interpolate them.

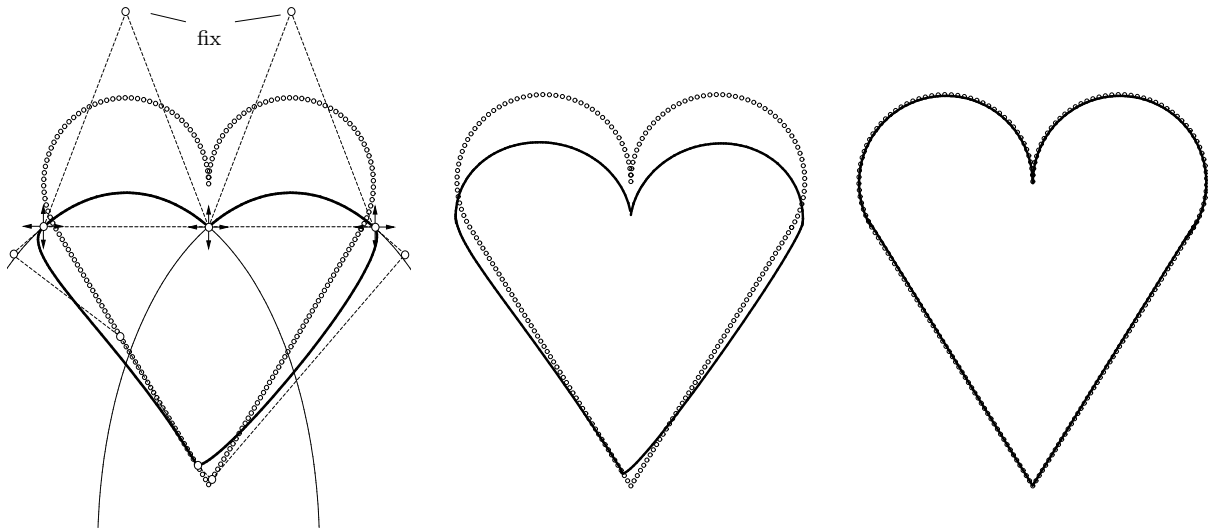


Figure 3: Fitting of a hybrid curve. Initial value (left), intermediate result after 5 iterations (center) and final result after 20 iterations (right).

The shape parameters are the control points of the parametric curves, the remaining 2×3 coefficients of the quadratic polynomials defining the conics, and the coordinates of the lower two vertices of the two basis triangles (but not the top ones, which are fixed.). In addition, we require continuity everywhere and tangent continuity between conics and parametric curves, and use the resulting conditions to eliminate some of the shape parameters. (E.g., the two inner vertices of the two basis triangles are made to be equal, etc.) Summing up, this leads to a vector \mathbf{s} containing 20 shape parameters.

The approximating curve is generated by the following algorithm.

1. Initialization: Find initial shape parameters $\mathbf{s}(0)$. One may start with a user-defined curve, or find an initial curve by picking the best curve among a set of curves obtained by randomly sampling the shape parameters.
2. Compute the closest points of the given data points on the current curve and define the associated normal velocities.
3. Find the derivatives $\dot{\mathbf{s}}(t)$ of the shape parameters by solving the linear system (5).
4. Choose the step size $h = \min(1, \{C/v(u_j)\}_{j=1,\dots,N})$, where the user-defined constant C specifies the largest tolerated displacement of a closest point on the curve, and update the shape parameters, $\mathbf{s}(t+h) = \mathbf{s}(t) + h\dot{\mathbf{s}}(t)$.
5. The algorithm terminates if $\|\dot{\mathbf{s}}(t)\|$ is below a user-defined threshold; otherwise continue with step 2.

Figure 3 shows the evolution of this hybrid curve for approximating 200 points sampled from a heart-like shape. Note that the hybrid model automatically detects the joints between linear and circular segments in the given data. Clearly, this is only possible by exploiting the additional a-priori information about the target shape.

5 Concluding remarks

We introduced an abstract framework for fitting curves to unorganized data by a true evolution process, which generalizes the technique of normal (or tangent) distance minimization. This technique can deal with a large class of curve representations, which even

includes hybrid objects. This makes it possible to exploit additional a-priori information about the geometry of the target object. In addition, one may easily extend the procedure to other velocity fields, which may, e.g., be derived by combining information from images with curvature information.

Future research will address the generalization to objects in 3-dimensional space and the characterization of stationary points in the general case. Also, in the case of noisy data, we need to study the problem of how to associated the points with the curve in more detail. In the case of sharp vertices, some of the points may end up on the wrong branch, leading to a relatively large error in the position of the vertex, see Figure 2.

Acknowledgment. The authors were supported by the Austrian Science Fund (FWF) through the FSP S 92 “Industrial Geometry”.

References

- [1] M. Aigner and B. Jüttler, Robust Computation of Foot Points on Implicitly Defined Curves, in: *Mathematical Methods for Curves and Surfaces: Tromsø 2004* (M. Dæhlen et al., eds.), Nashboro Press, 2005, 1–10.
- [2] M. Alhanaty and M. Bercovier, Curve and surface fitting and design by optimal control methods, *Computer-Aided Design* 33 (2001), 167–182
- [3] A. Blake and M. Isard, *Active contours*, Springer, 2000.
- [4] H. Hagen, G. Brunnett and P. Santarelli, Variational Principles in Curve and Surface Design, *Surv. Math. Ind.* 3, 1-27, 1993.
- [5] K. E. Hoff et al., Fast computation of generalized Voronoi diagrams using graphics hardware, *SIG-GRAPH'99 proceedings*, 277–286.
- [6] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, AK Peters, Wellesley Mass., 1996.
- [7] S.-M. Hu and J. Wallner, A second order algorithm for orthogonal projection onto curves and surfaces, *Comp. Aided Geom. Des.* 22 (2005), 251–260.
- [8] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer, 1999.
- [9] S. Osher and J. Sethian, Fronts propagating with curvature dependent speed, algorithms based on a Hamilton-Jacobi formulation, *J. Comp. Phys.* 79 (1988), 12–49.
- [10] S. Osher and R. P. Fedkiw, *Level set methods and dynamic implicit surfaces*, Springer, 2003
- [11] D. F. Rogers and N. G. Fog, Constrained B-spline curve and surface fitting, *Computer Aided Design* 21 (1989), 641–648.
- [12] B. Sarkar and C.-H. Menq, Parameter optimization in approximating curves and surfaces to measurement data, *Comp. Aided Geom. Design* 8 (1991), 267–280
- [13] H. Pottmann and S. Leopoldseder, A concept for parametric surface fitting which avoids the parametrization problem. *Comp. Aided Geom. Design* 20 (2003), 343-362.
- [14] H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active B-spline curves and surfaces. *Proc. Pacific Graphics 2002*, IEEE Press, 8–25.
- [15] H. Pottmann et al., Industrial geometry: recent advances and applications in CAD, *Computer-Aided Design* 37 (2005), 751–766.
- [16] N.J. Redding, Implicit polynomials, orthogonal distance regression and the closest point on a curve, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 22 (2000), 191–199.
- [17] T. Speer, M. Kuppe, and J. Hoschek, Global reparametrization for curve approximation, *Comput. Aided Geom. Design* 15 (1998), 869–877.
- [18] M. Kass, A. Witkin and D. Terzopoulos, Snakes: active contour models. *Int. J. Comp. Vision* 1.4 (1987), 321–331.
- [19] W. Wang, H. Pottmann and Y. Liu, Fitting B-spline curves to point clouds by squared distance minimization. *ACM Transactions on Graphics*, in press.
- [20] H. Yang et al., Evolution of T-spline Level Sets with Distance Field Constraints for Geometry Reconstruction and Image Segmentation, submitted, FSP report no. 1 (2005), available at www.ig.jku.at.