

# Reconstruction and smoothing of polygonal curves

*Tim Volodine*

*Denis Vanderstraeten*

*Dirk Roose*

*Report TW433, June 2005*

Katholieke Universiteit Leuven  
Department of Computer Science  
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# Reconstruction and smoothing of polygonal curves

*Tim Volodine*

*Denis Vanderstraeten*

*Dirk Roose*

*Report TW 433, June 2005*

Department of Computer Science, K.U.Leuven

## **Abstract**

In this paper we propose a method for piecewise linear reconstruction and subsequent smoothing of a point sampled curve. The reconstruction step is based on the meshless parameterization reconstruction algorithm proposed by Floater. The information computed in the reconstruction step is used for a least squares based discrete smoothing method, with behavior comparable to a smoothing spline. We discuss smoothing under two different constraints: one based on the sum of squared deviations and the other based on a max-norm error. The smoothing algorithm is essentially a variation on the least squares meshes concept proposed by Sorkine et al. The reconstruction and smoothing steps are conceptually similar and provide a unified approach to the reconstruction of smooth curves in an arbitrary dimensional space.

**Keywords :** Line and curve generation, curve reconstruction, least squares approximation.

**CR Subject Classification :** I.3.3, G.1.2.

# Reconstruction and smoothing of polygonal curves

T. Volodine\*, D. Vanderstraeten and D. Roose

Katholieke Universiteit Leuven  
Metris N.V. Belgium

June 18, 2005

## Abstract

In this paper we propose a method for piecewise linear reconstruction and subsequent smoothing of a point sampled curve. The reconstruction step is based on the meshless parameterization reconstruction algorithm proposed by Floater [Flo03]. The information computed in the reconstruction step is used for a least squares based discrete smoothing method, with behavior comparable to a smoothing spline. We discuss smoothing under two different constraints: one based on the sum of squared deviations and the other based on a max-norm error. The smoothing algorithm is essentially a variation on the least squares meshes concept proposed by Sorkine et al. [SCO04]. The reconstruction and smoothing steps are conceptually similar and provide a unified approach to the reconstruction of smooth curves in an arbitrary dimensional space.

## 1 Introduction

The *curve reconstruction* problem, where a curve has to be reconstructed from a given unorganized point sample, is extensively studied in the domain of computational geometry and reverse engineering. The reconstructed curve can be either piecewise linear or have some continuity imposed on its derivatives, resulting in a smooth curve. In this paper we are concerned with the *polygonal curve reconstruction*, i.e. the output is a piecewise linear curve connecting all the sample points.

In the literature, most of the proposed algorithms use concepts from computational geometry and combinatorial optimization. In [DMR99], for example, a *De-launay triangulation* and a *Voronoi diagram* are used to obtain a proper ordering

---

\*e-mail: timv@cs.kuleuven.be

of the sample points. Other concepts like the  $\alpha$ -shapes [EKS83, BB97], *crusts* and  $\beta$ -*skeletons* [ABE98], which are tightly related to the Delaunay triangulation, were successfully applied to the reconstruction problem, for an overview see [Ede98]. In [Gie99] Giesen proposed a different *combinatorial* approach. He showed that, under certain sampling conditions, the curve reconstruction problem can be solved by constructing a *traveling salesman path*. This can be done in polynomial time, provided a suitable sampling condition is satisfied, as explained in [AM00]. There are also approaches involving the use of the *minimal spanning tree* [dG95, Lee00].

Another reconstruction algorithm of more numerical nature is due to Floater [Flo03]. It is called *meshless parameterization*, because it maps the sample points on the real line, by solving a linear system, to obtain an ordering of the points. This method forms the basis of the reconstruction algorithm in this paper.

There is a large amount of literature on the problem of curve smoothing as well. While most of the literature is concerned with some *continuously differentiable*  $C^k$  approximations, like splines or radial basis functions [dB78, Die95, CBM\*03], the *discrete methods* [Tau95, SCO04, Lee00] constitute another approach, which is attractive because of its simplicity and efficiency. In [Tau95] for example, Taubin proposed an improved discrete Laplacian smoothing algorithm, while [Lee00] uses a moving least squares method to smooth a noisy point sampled curve. The smoothing method presented in this paper is also discrete, in the sense that it transforms a polygonal curve into a smoother one. The algorithm is inspired by [SCO04] and uses notions from the *smoothing spline* literature.

## 2 Meshless curve reconstruction

In [Flo03] Floater proposed a method for the reconstruction of a piecewise linear approximation of a smooth curve from its point samples  $\{\mathbf{p}_i\}_1^n$  in  $\mathbb{R}^s$ . Provided the two end points of the curve are known, the method computes the parameterization of the points on the real line. It assigns to each sample point  $\mathbf{p}_i \in \mathbb{R}^s$  a parameter point  $t_i \in \mathbb{R}$ , by requiring each *interior* parameter point  $t_i$  to be a convex combination of its neighboring parameter points, i.e.

$$\left( \sum_{j \in \mathcal{I}_i} w_{ij} \right) t_i = \sum_{j \in \mathcal{I}_i} w_{ij} t_j, \quad w_{ij} \geq 0, \quad (1)$$

where  $\mathcal{I}_i$  is the index set of the neighbors of  $\mathbf{p}_i$ . Assume for a moment that *all*  $t_i$  correspond to the interior points. Then, the equations in (1) can be written as  $W\mathbf{t} = \mathbf{0}$  where  $W \in \mathbb{R}^{n \times n}$  is the *general unnormalized Laplacian matrix*, containing  $-w_{ij}$  as off-diagonal entries ( $w_{ij}$  is zero when points  $i$  and  $j$  are not neighbors) and their row-sum on the diagonal.

For convenience of notation, we define  $F_k$  as the matrix derived from  $W$  by replacing the first  $k$  rows with a  $k$  by  $k$  unity matrix augmented with zeros

$$F_k = \left[ \begin{array}{c|c} \mathbf{I}_{k \times k} & \mathbf{0} \\ \hline \cdot & \cdot \end{array} \right]. \quad (2)$$

The dots denote the remaining entries from the matrix  $W$ .

Without loss of generality, let  $\mathbf{p}_1$  and  $\mathbf{p}_2$  be the two endpoints of the curve. The method of Floater proceeds by fixing the corresponding parameter values  $t_1$  and  $t_2$  on the real line by setting  $t_1 = 0$  and  $t_2 = 1$ . The interior parameter points  $\mathbf{t}' = [t_3 \dots t_n]^\top$  are then obtained by solving the linear system

$$F_2 \begin{bmatrix} t_1 & t_2 & \mathbf{t}' \end{bmatrix}^\top = \begin{bmatrix} 0 & 1 & \mathbf{0}_{1 \times n-2} \end{bmatrix}^\top. \quad (3)$$

By sorting these  $t_i$  values we obtain an ordering of the original sample points  $\mathbf{p}_i$ , which is the solution to the reconstruction problem.

## 2.1 Meshless parameterization of closed curves

The meshless parameterization method of Floater works only for open curves, i.e. it requires the two endpoints of the curve to be available. Therefore, when the points  $\{\mathbf{p}_i\}_1^n$  are sampled from a closed curve, this method cannot be applied directly. This problem can be solved by partitioning the sample in two or more samples and reconstructing them separately [FR01, VRV03]. However, often it is not obvious how to construct such partition. The problem of partitioning a sample is illustrated in figure 1. In 1(a) we see the typical problems of the nearest neighbor approach (NN-CRUST) from [DK99] near sharp corners. Figure 1(b) shows the result of the reconstruction algorithm from [VRV03] with a badly chosen seed point. This kind of behaviour is typical for any region growing partitioning algorithm based on nearest neighbor relation. For this reason we propose an algorithm which does not perform any explicit partitioning in the sample space and which can be seen as a natural extension of the meshless parameterization algorithm for open curves, see figure 1(c).

In our approach the parameterization for closed curves is computed in the plane instead of on the real line. The idea is to produce a square-like shape, by fixing 4 parameter points in the plane, as shown in figure 2. In fact the algorithm can be viewed as a discrete analogue of stretching a rubber sheet. The sheet is modelled as a mass-spring system, where between some points (masses) a spring is placed with some stiffness coefficient ( $w_{ij}$  in (1)), depending on the Euclidean distance between them.

With each sample point  $\mathbf{p}_i$  we associate a point  $z_i$  in the plane, represented by a complex number. We use complex numbers for simplicity of notation. In practice,

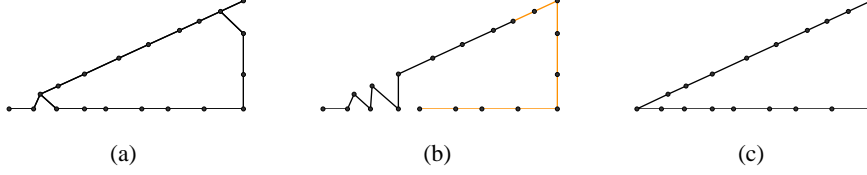


Figure 1: (a) reconstruction using the NN-CRUST algorithm (b) reconstruction by the region growing partitioning algorithm [VRV03] (c) reconstruction by the proposed 2D meshless parameterization

each system written with complex numbers amounts to solving *two* real systems, one for the *real* and one for the *imaginary* component.

### 2.1.1 Selection of fixed points

For the selection of the four fixed parameter points we adopt the following heuristic. First we choose two farthest points in the sample  $\mathbf{p}_{i_1}$  and  $\mathbf{p}_{i_2}$ . Without losing generality we assume that they have indices  $i_1 = 1$  and  $i_2 = 2$ . For the ease of notation we denote the other two fixed parameter points by  $z_3$  and  $z_4$ . Starting from the *complex* version of system (1), i.e.  $W\mathbf{z} = 0$ , we solve the resulting system

$$F_2 \begin{bmatrix} z_1 & z_2 & \mathbf{z}^{(1)} \end{bmatrix}^\top = \begin{bmatrix} -1 & 1 & \mathbf{0}_{1 \times n-2} \end{bmatrix}^\top, \quad (4)$$

yielding the solution  $\mathbf{z}^{(1)} \in \mathbb{C}^{n-2}$ . From this solution, the parameter point  $z_3$  is chosen, such that it is the closest point to the origin, i.e.  $z_3 = \arg \min |z_i^{(1)}|$ . Now, with three points  $z_1, z_2$  and  $z_3$  kept fixed we solve

$$F_3 \begin{bmatrix} z_1 & z_2 & z_3 & \mathbf{z}^{(2)} \end{bmatrix}^\top = \begin{bmatrix} -1 & 1 & i & \mathbf{0}_{1 \times n-3} \end{bmatrix}^\top, \quad (5)$$

where  $\mathbf{z}^{(2)} \in \mathbb{C}^{n-3}$  and  $i$  is the imaginary unit. The point  $z_4$  is then chosen analogously, as the point with the smallest modulus in  $\mathbf{z}^{(2)}$ . Finally, to obtain the 4 fixed points solution we solve

$$F_4 \begin{bmatrix} z_1 & z_2 & z_3 & z_4 & \mathbf{z}^{(3)} \end{bmatrix}^\top = \begin{bmatrix} -1 & 1 & i & -i & \mathbf{0}_{1 \times n-4} \end{bmatrix}^\top. \quad (6)$$

To illustrate this procedure, four snapshots of the algorithm are shown in figure 2. In the first figure 2(a) the solution of (4) is shown. The point  $z_3$  is the encircled point closest to the origin. Figure 2(b) shows the solution of (5), with the encircled point  $z_4$ . The solution where all four points are fixed can be seen in figure 2(c).

The choice of the fixed points appears to be important, because a bad choice leads to greater distortion of the square shape, increasing the risk of an incorrect reconstruction. The motivation of the choice of  $z_3$  and  $z_4$  in our approach is that in case of near-uniform sampling of the points  $\mathbf{p}_i$  this choice yields a solution with minimal distortion, due to symmetry considerations.

### 2.1.2 Partitioning in the plane

To obtain an ordering of the points we could sort the obtained parameter points according to the polar angle. However, sometimes the reconstruction is wrong near the fixed vertices, because there the largest distortion occurs. This phenomenon can be seen in figure 2(c) in the vicinity of  $z_1$ . To improve the reconstruction we partition the parameter points in four disjoint sets, and solve a modified system (6). In this way we obtain a solution in which all the parameter points lie on the square, these points can then be ordered by their polar angle (see figure 2(d)).

Define  $angle(z) = atan2(\Im z/\Re z)$ , where  $atan2$  is the four-quadrant arctangent function, returning angles in  $[-\pi, \pi]$ . Consider in each quadrant, the point closest to the bisector in terms of angle, i.e. in this way we obtain 4 points  $\{z_{i_1}, z_{i_2}, z_{i_3}, z_{i_4}\}$  with index

$$i_k = \arg \min_i |angle(z_i) - \alpha_k|, \quad (7)$$

where  $\alpha_1 = \pi/4$ ,  $\alpha_2 = 3\pi/4$ ,  $\alpha_3 = -3\pi/4$  and  $\alpha_4 = -\pi/4$ . The parameter points are separated in four sets  $I_1, \bar{I}_1, I_2$  and  $\bar{I}_2$  with the separating points  $z_{i_k}, k = 1..4$  (figure 2(c)). These sets are subsets of  $I = \{1, 2, \dots, n\}$  and are defined as follows

$$\begin{aligned} I_1 &= \{i \mid angle(z_i) > angle(z_{i_3}) \text{ and } angle(z_i) < angle(z_{i_1})\} \\ \bar{I}_1 &= I \setminus \{I_1 \cup \{i_3, i_1\}\} \\ I_2 &= \{i \mid angle(z_i) > angle(z_{i_4}) \text{ and } angle(z_i) < angle(z_{i_2})\} \\ \bar{I}_2 &= I \setminus \{I_2 \cup \{i_4, i_2\}\}. \end{aligned} \quad (8)$$

To obtain a solution lying entirely on the square it is necessary that points in different quadrants are not connected to each other, i.e. the weights between them are zero. This can be achieved by zeroing all the elements of  $F_4$ ,  $f_{ij}$  or  $f_{ji}$  for which  $i \in I_1, j \in \bar{I}_1$  or  $i \in I_2, j \in \bar{I}_2$ . To preserve the convex combination formulation, the diagonal entries are also modified, such that  $f_{ii} = -\sum_{j \neq i} f_{ij}$ . We will denote the modified matrix  $F_4$  by  $\tilde{F}_4$ . Solving system

$$\tilde{F}_4 \begin{bmatrix} z_{i_1} & z_{i_2} & z_{i_3} & z_{i_4} & \mathbf{z}^{(4)} \end{bmatrix}^\top = \begin{bmatrix} 1 & i & -1 & -i & \mathbf{0}_{1 \times n-4} \end{bmatrix}^\top, \quad (9)$$

yields a solution lying on the square as in figure 2(d). The ordering, and therefore the reconstruction, is then obtained by sorting these points according to their *angle* w.r.t. the origin.

### 2.1.3 Solution of the systems

When solving (4) we know that  $\Im \mathbf{z}^{(1)} = 0$ . Hence it is sufficient to solve one real system corresponding to the real part of the solution. System (5) amounts to solving two real systems, where only the right hand side is different. The solution to both systems can be computed efficiently using *LU*-factorization of  $F_3$ , reducing the solution cost of (5) approximately to the solution of one real system. The same applies to (6). Solution of (9) can be obtained by solving four smaller real systems, one for each side of the square. Therefore, the total computational cost is approximately equal to the work required to solve *four*  $n \times n$  linear systems. Because the involved matrices are very sparse and have small effective bandwidth, dedicated sparse direct solvers like UMFPACK and SuperLU are very efficient and can solve sparse systems with thousands of unknowns in under a second [VRV03].

## 2.2 On the choice of neighborhoods and weights

The two essential choices in the *meshless parameterization* approach are the selection of neighbors and the choice of weights. The two most popular choices for neighborhoods are the *ball neighborhood* and the *k-nearest neighbors* neighborhood. The first choice is more suitable for non-uniform samples, while the second is generally applicable to more or less uniform samples.

### 2.2.1 Construction of weights

A convenient way to construct the weights  $w_{ij}$  is to use a radial function  $\psi$ , defined on  $[0, 1]$ . Given a sample point  $\mathbf{p}_i$  and its neighborhood  $\mathcal{N}_i$ , the weights are then computed as

$$w_{ij} = \psi \left( \frac{\|\mathbf{p}_j - \mathbf{p}_i\|}{R_{max}} \right), \quad R_{max} = \max_{\mathbf{p}_j \in \mathcal{N}_i} \|\mathbf{p}_j - \mathbf{p}_i\|. \quad (10)$$

In [Flo03] it was shown that any function  $\psi(r)$  can be used, as long as it is positive and monotonically decreasing in  $[0, 1]$ . In most examples in this paper the *k*-nearest neighbors were used with  $k = 6$  to 15 points and  $\psi(r) = e^{-r}$ . Other choices for  $\psi(r)$  are discussed in section 5.2.

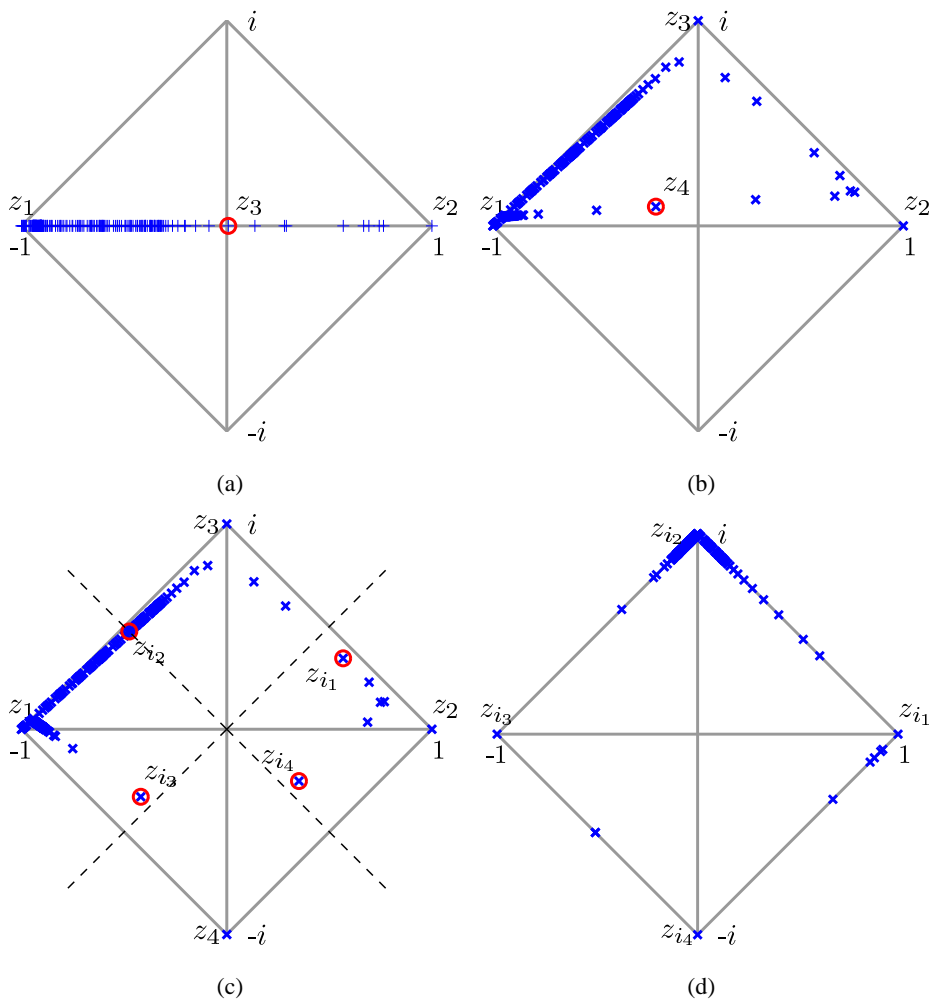


Figure 2: Situation after solving system (a) (4) (b) (5) (c) (6) (d) (9)

### 2.2.2 Choice of the neighborhoods

In order for the matrices  $F_2$ ,  $F_3$ ,  $F_4$  and  $\tilde{F}_4$  to be non-singular it is important to choose the neighborhoods large enough. In the next paragraphs we describe efficient pre-processing methods to insure a correct choice of the size of the neighborhoods, before executing the reconstruction algorithm. For this purpose it is convenient to consider a graph theoretic interpretation of the matrices involved.

Consider the directed graph  $G(F)$  with  $n$  vertices  $v_i$  corresponding to the  $n \times n$  matrix  $F = (f_{ij})$ , where there is a directed edge between each two vertices  $v_i$  and  $v_j$  iff  $f_{ij} \neq 0$ . We will call a non-fixed parameter point  $z_i$  *boundary connected* if there is a path in  $G(F)$ , i.e.  $v_{i_1}, v_{i_2}, \dots, v_{i_k}$ , where  $v_{i_k}$  is a vertex corresponding to a *fixed* parameter point  $z_{i_k}$ , e.g. if  $F = F_2$  then  $v_{i_k}$  is  $v_1$  or  $v_2$ . Furthermore, a graph  $G(F)$  in which there is a path between each two vertices is called *strongly connected*, and the matrix  $F$  is called *irreducible*.

A condition for  $F_2$  to be non-singular, given in [FR01], states that it is sufficient for all non-fixed (interior) vertices of  $G(F_2)$  to be boundary connected. However, non-singularity of  $F_2$  does not guarantee a correct reconstruction. This guarantee is provided by the sufficient conditions given in [Flo03]. Besides requiring that the neighborhood of each sample point contains its two immediate neighbors in the correct reconstruction, the conditions impose requirements on the radii of the neighborhoods. Roughly speaking the original curve may not ‘turn back’, in the sense that for each two *successive* points  $\mathbf{p}_{i_k}$  and  $\mathbf{p}_{i_{k+1}}$  on the curve there is no other point  $\mathbf{p}_{i_r} \in \mathcal{N}_{i_k}$  ( $r > k + 1$ ) closer to  $\mathbf{p}_{i_k}$  than to  $\mathbf{p}_{i_{k+1}}$ .

If these conditions are satisfied then the generalized Laplacian matrix  $W$  is irreducible. Hence to determine whether the chosen neighborhoods are large enough, we might simply check whether  $G(W)$  is strongly connected. For this purpose there exist *linear* time algorithms, running in  $O(n + e)$ , with  $n$  the number of vertices and  $e$  the number of edges in the graph. Assuming the neighborhoods are not too large,  $e$  is  $O(n)$ , which justifies the term *linear*.

When using ball neighborhoods the matrix  $W$  is symmetric and the corresponding graph  $G(W)$  can be viewed as an *undirected* graph. The connectivity is then easily checked by means of a breadth-first traversal of  $G(W)$ . With an efficient implementation this can be done in  $O(n + e)$  steps [Sed02].

In the case of neighborhoods based on the  $k$ -nearest neighbors,  $W$  is generally not symmetric, and  $G(W)$  is a *directed* graph. In this case various efficient algorithms are available, e.g. algorithms of Tarjan, Kosaraju and Gabow compute the strong components of a graph with time complexity of  $O(n + e)$  [Sed02].

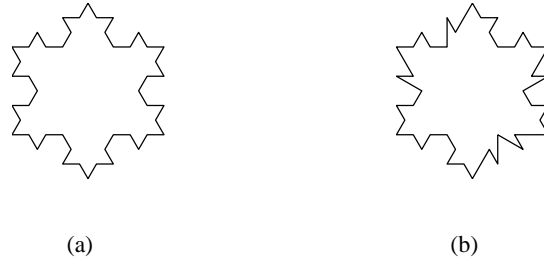


Figure 3: (a) original Koch snowflake fractal (b) (incorrect) reconstruction

### 2.2.3 Final solution

Note that to determine whether the neighborhoods are well chosen, it is sufficient to check the irreducibility of the matrix  $W$ . If  $W$  is irreducible, the matrices  $F_2$ ,  $F_3$ ,  $F_4$  and  $\tilde{F}_4$  will be non-singular.

The irreducibility of  $W$  however, is not sufficient to guarantee that the solution will be non-degenerate. Imagine a branch in  $G(W)$  which is connected to only one vertex. This branch will collapse into that vertex in the solution, yielding duplicate parameter values. Therefore it remains necessary to check duplicates in the final solution.

To illustrate that the behavior of our closed curve algorithm is similar to the one proposed by Floater for open curves, we use a limit case shown in 3(a). This is an example of the Koch snowflake where the angles in the polygon are  $60^\circ$  and distances to the neighboring points are equal. Obviously, for any choice of neighborhoods, the conditions given in [Flo03] are not satisfied, resulting in an incorrect reconstruction, see figure 3(b).

## 3 Smoothing with constraints on the sum of squared errors

In this and following sections we show how to obtain a smoother version of the original polygonal curve  $\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \dots, \mathbf{p}_{i_n}$  by computing new points  $\mathbf{x}_i$ , which satisfy the *convex combination* requirements *and* some constraints on their position. This smoothing procedure is equivalent to the approach presented in [SCO04]. However, our derivation and application of the method is different. Because of the nature of the constraints we will call the method described in this section (discrete) *SSE-smoothing*. In the next section we will use different constraints, which yield the so-called *max-norm smoothing* method.

### 3.1 Smoothing of closed curves

Let  $\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \dots, \mathbf{p}_{i_n}$  be the vertices of the polygonal reconstruction of a curve through the sample points  $\mathbf{p}_i$ , as shown on figure 4(a). We can view this reconstruction as a graph representation of a physical system where each vertex has unit mass and each edge represents a spring with stiffness  $\kappa_i$  and zero rest length. The requirement for each sample point to be a *convex combination* of its two immediate neighbors, or equivalently that the magnitudes of the forces exerted by the two springs on  $\mathbf{p}_i$  are equal, can be expressed as

$$A\mathbf{x} = 0, \quad (11)$$

where  $A$  is a symmetrical positive semidefinite matrix

$$A = \begin{bmatrix} \kappa_1 + \kappa_5 & -\kappa_1 & & & -\kappa_5 \\ -\kappa_1 & \kappa_1 + \kappa_2 & -\kappa_2 & & \\ & -\kappa_2 & \kappa_2 + \kappa_3 & -\kappa_3 & \\ & & -\kappa_3 & \kappa_3 + \kappa_4 & -\kappa_4 \\ -\kappa_5 & & & -\kappa_4 & \kappa_4 + \kappa_5 \end{bmatrix}. \quad (12)$$

This matrix can be directly obtained from  $W$  using a permutation  $\sigma$  of the original points  $\mathbf{p}_i$  such that  $\{p_{\sigma(i)}\}_{i=1}^n$  is the polygonal reconstruction. In further discussion, without losing generality, we assume that  $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n$  is the polygonal reconstruction.

Let  $p_i^{(k)}$  be the  $k^{\text{th}}$  component of point  $\mathbf{p}_i \in \mathbb{R}^s$ , i.e.  $\mathbf{p}_i = (p_i^{(1)}, p_i^{(2)}, \dots, p_i^{(s)})^\top$ . Let  $\mathbf{x}_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(s)})^\top$  be the positions of points  $\mathbf{p}_i$  after smoothing. The idea is to find  $\mathbf{x}_i$  such that the deviation from the *equilibrium condition* (11) is minimized in the least squares sense, without deviating too much from the original positions of the sample points. By letting  $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})^\top$ , we can write this requirement as

$$\text{minimize } \sum_{k=1..s} \|A\mathbf{x}^{(k)}\|_2^2, \text{ s.t. } \sum_{i=1..n} \omega_i^2 \|\mathbf{p}_i - \mathbf{x}_i\|_2^2 \leq \tau, \quad (13)$$

where  $\omega_i$  are user-specified weights assigned to each sample point  $\mathbf{p}_i$  and  $\tau$  is the smoothing factor. To solve this problem, we use the *method of weighting* [GL96]. Let  $D = \text{diag}(\omega_1, \omega_2, \dots, \omega_n)$ . For each dimension  $k$ , we write  $\mathbf{d}^{(k)} = D(p_1^{(k)}, p_2^{(k)}, \dots, p_n^{(k)})^\top$ . The solution to (13) is obtained independently in each dimension, by solving the following least squares problem

$$\min_{\mathbf{x}^{(k)}} \left\| \begin{bmatrix} A \\ \sqrt{\lambda}D \end{bmatrix} \mathbf{x}^{(k)} - \begin{bmatrix} \mathbf{0}_{n \times 1} \\ \sqrt{\lambda} \mathbf{d}^{(k)} \end{bmatrix} \right\|_2. \quad (14)$$

The parameter  $\lambda$  determines the importance of the constraint that the new points remain on their positions. Obviously,  $\lambda = 0$  yields a trivial zero solution, meaning the potential energy of the system is minimized without any restriction. Setting  $\lambda = \infty$  results in the original sample points. By varying  $\lambda$  between these two extremes it is possible to find a value which satisfies (13) for a pre-specified  $\tau$ . To determine the optimal value define the deviation function

$$\phi^{(k)}(\lambda) = \|D\mathbf{x}^{(k)}(\lambda) - \mathbf{d}^{(k)}\|_2^2, \quad (15)$$

where  $\mathbf{x}^{(k)}(\lambda)$  is the solution of (14) for a given  $\lambda$ . By summing over all dimensions we obtain the *total* deviation function

$$\phi(\lambda) = \sum_{k=1}^d \phi^{(k)}(\lambda). \quad (16)$$

To solve (13) it is sufficient to find a  $\lambda^*$  for which  $\phi(\lambda^*) = \tau$ .

It is well known from [GL96, , chapter 12] that each deviation function  $\phi^{(k)}(\lambda)$  can be written in terms of the general singular decomposition of  $A$  and  $D$ . Let

$$U^\top AX = \text{diag}(\alpha_1, \alpha_2, \dots, \alpha_n) \quad (17)$$

$$V^\top DX = \text{diag}(\beta_1, \beta_2, \dots, \beta_n) \quad (18)$$

with  $\alpha_i \geq 0$ ,  $\beta_i \geq 0$ , square unitary matrices  $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n]$  and  $V = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n]$  and a non-singular square matrix  $X$ . In our particular case we then have

$$\phi^{(k)}(\lambda) = \sum_{i=1}^n \left( \frac{\alpha_i^2 \mathbf{v}_i^\top \mathbf{d}^{(k)}}{\alpha_i^2 + \lambda \beta_i^2} \right)^2 \quad (19)$$

This function is strictly decreasing for positive  $\lambda$  and strictly convex. The sum  $\phi(\lambda)$  of these functions also possesses these properties. Therefore, to find the optimal value  $\lambda^*$  it is possible to use the Newton-Raphson algorithm, which is guaranteed to converge for any starting value  $\lambda_0 \in [0, \lambda^*]$ . However, there is a faster and easier way to obtain a sufficiently optimal  $\lambda^*$  by means of a rational interpolation scheme, as described by Dierckx in [Die95]. This method does not require the computation of the generalized singular value decomposition, only a least squares solver is required. For smoothing purposes it is not necessary to find  $\lambda^*$  to full precision. The function  $\phi(\lambda)$  flattens very rapidly (its first derivative rapidly tends to zero) and therefore a small deviation from the optimum will be indistinguishable in visualization applications.

We briefly outline the adopted interpolation scheme, further details can be found in [Die95]. The proposed rational interpolation scheme is an iterative method

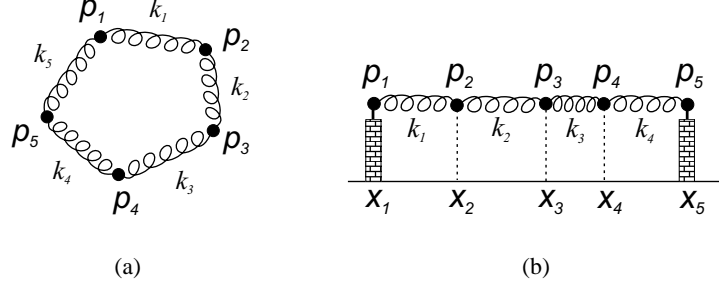


Figure 4: mass-spring system: (a) closed case (b) non-closed case

which starts with three values  $\lambda_1 = 0$ ,  $\lambda_2 = 1$ ,  $\lambda_3 = \infty$  and the corresponding values of the error function  $\phi(\lambda)$ ,  $f_1 = \phi(\lambda_1)$ ,  $f_2 = \phi(\lambda_2)$ ,  $f_3 = 0$ . Through these points the rational interpolating function  $R(\lambda) = (u\lambda + v)/(\lambda + w)$  is computed. Requiring  $R(\lambda) = \tau$  yields a better estimate  $\tilde{\lambda}$ , which is used in the next iteration. The algorithm terminates when  $|\phi(\tilde{\lambda}) - \tau| \leq \tau\xi$ , where  $\xi$  is the tolerance and is usually equal to  $1e - 3$ . In the examples presented in this paper 5 or 6 iterations were sufficient.

### 3.2 Smoothing of non-closed curves

In case the sample points are obtained from a non-closed curve we can apply a similar smoothing algorithm. Often it is desirable to have interpolation at the end points of the curve, hence after imposing these conditions on (12), we obtain a similar matrix  $A_c$

$$A_c = \begin{bmatrix} \kappa_1 + \kappa_2 & -\kappa_2 & & & \\ -\kappa_2 & \kappa_2 + \kappa_3 & -\kappa_3 & & \\ & -\kappa_3 & \kappa_3 + \kappa_4 & & \\ & & & \kappa_4 + \kappa_5 & \\ & & & & \kappa_5 \end{bmatrix}. \quad (20)$$

This symmetric Jacobi matrix describes a chain of masses coupled via springs and fixed at both end points as in figure 4(b). The solution of

$$A_c \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \kappa_1 x_1 \\ 0 \\ \kappa_4 x_5 \end{bmatrix} \quad (21)$$

yields the equilibrium position of the free unit masses at positions  $\mathbf{p}_2$ ,  $\mathbf{p}_3$  and  $\mathbf{p}_4$ , minimizing the potential energy of the mass-spring system. Contrary to the closed curve case the points  $\mathbf{p}_1$  and  $\mathbf{p}_5$  are fixed, hence the right hand side of the above equation is not zero. However the error function  $\phi^{(k)}(\lambda)$  retains its properties of

being strictly convex and decreasing [GL96]. Therefore it is possible to use the outlined algorithm without further modification.

### 3.3 On the solution of the least squares system

For the computation of the parameter  $\lambda$  in (14) we need to solve  $s$  least squares systems per iteration step. The system for the  $k^{\text{th}}$  component ( $k = 1..s$ )

$$\begin{bmatrix} A \\ \sqrt{\lambda}D \end{bmatrix} \mathbf{x}^{(k)} = \begin{bmatrix} \mathbf{b} \\ \sqrt{\lambda}\mathbf{d}^{(k)} \end{bmatrix} \quad (22)$$

is very structured and sparse. Similar sparse least squares systems arise in data fitting applications, such as polynomial spline fitting and smoothing [dB78, Die95]. Such systems can be solved efficiently by using Givens rotations to compute the  $QR$  factorization of the original matrix [Cox81]. In our case the specific structure of system (22) allows us to compute the solution  $\mathbf{x}^{(k)}$  in *linear* time and space. Below we show how this can be done when the matrix  $A$  is tridiagonal, i.e. corresponding to the non-closed curve case with interpolation conditions. Analogous reasoning can be applied to the closed curve case (12). In that case the computation is slightly more expensive, but remains of *linear* complexity.

Let  $A$  be the  $n \times n$  matrix, as in (20). This matrix is tridiagonal, hence it can be orthogonally transformed into an upper triangular  $\tilde{R}$  with bandwidth equal to three. This transformation can be performed in linear time because it requires  $n - 1$  Givens rotations (one for each row except the first one) and each Givens rotation has to update only 3 elements per row ( $A$  is tridiagonal). By applying appropriate Givens rotations we obtain the  $QR$  decomposition of  $A$

$$A = \tilde{Q}\tilde{R}, \quad \tilde{Q}, \tilde{R} \in \mathbb{R}^{n \times n}. \quad (23)$$

This decomposition only needs to be computed once, because  $A$  does not change with varying  $\lambda$ . To solve (22) we can equivalently solve

$$\begin{bmatrix} \tilde{R} \\ \sqrt{\lambda}D \end{bmatrix} \mathbf{x}^{(k)} = \begin{bmatrix} \tilde{Q}^T \mathbf{b} \\ \sqrt{\lambda}\mathbf{d}^{(k)} \end{bmatrix}. \quad (24)$$

In general, consider a banded least squares system  $B\mathbf{x} = c$  with  $B$  a matrix of bandwidth  $\gamma$  with  $2n$  rows in *standard form*, meaning that the number of leading zero elements in a row is a non-decreasing function of the row number. Under these conditions it is shown in [Cox81] that the triangularization of  $B$  requires  $O(2n\gamma^2)$  multiplications. To bring (24) in *standard form* it is sufficient to interleave the rows of  $\sqrt{\lambda}D$  with those of  $\tilde{R}$ , as shown in figure 5. We write this permuted system as

$$Ex = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} \quad (25)$$

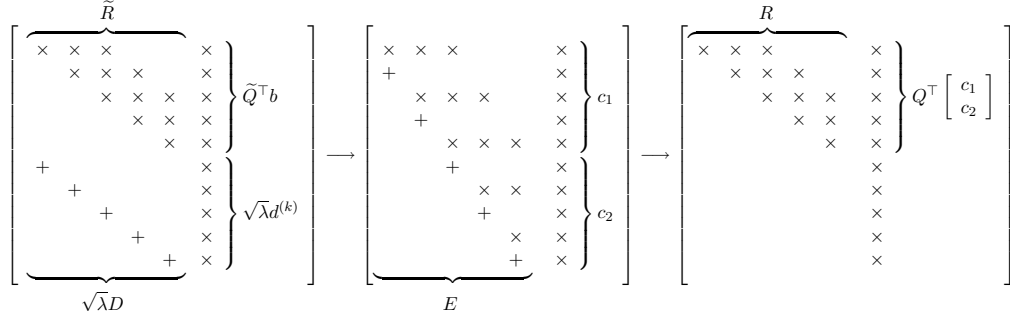


Figure 5: solving the least squares system: first the rows are interleaved to obtain a matrix  $E$  in *standard form*, second  $E$  is triangularized into  $R$  using Givens transformations.

with  $E \in \mathbb{R}^{2n \times n}$  and  $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^{n \times 1}$ . Let  $E = QR$  be the *reduced QR*-decomposition of  $E$ , width  $Q \in \mathbb{R}^{2n \times n}$  and  $R \in \mathbb{R}^{n \times n}$ . The solution is then obtained by solving a *banded* upper triangular system  $R\mathbf{x} = Q^T[\mathbf{c}_1 \ \mathbf{c}_2]^T$  by backsubstitution using  $O(2n\gamma)$  flops.

To avoid constructing the matrix  $Q$  and  $\tilde{Q}$  explicitly, the Givens rotations are applied to the right hand side directly. This can be done by augmenting  $E$  with the extra right hand side column, as shown in figure 5. In  $s$  dimensions the matrix  $E$  is augmented with  $s$  columns  $\mathbf{d}^{(1)}, \dots, \mathbf{d}^{(s)}$ , resulting in an extra cost of  $O(8n\gamma s)$  flops. When neglecting the computation cost associated with factorization of  $A$  and using *fast* Givens transformations, the *total* cost for the simultaneous solution of  $s$  systems for a given  $\lambda$  becomes  $O(2n\gamma(2\gamma + 5s))$ .

Because  $\tilde{R}$  and  $R$  have both bandwidth three, in the implementation we only need to store the three upper diagonals, requiring  $O(3n)$  memory usage.

### 3.4 Connection with smoothing splines

The SSE-smoothing method presented above resembles the smoothing spline approach widely studied in the literature [dB78, Die95]. The natural smoothing spline  $s_p$  of Schoenberg and Reinsch is constructed by finding a function  $s_p$  which minimizes

$$\lambda \sum_{i=1}^n w_i \|\mathbf{p}_i - s_p(u_i)\|_2^2 + \int_{u_1}^{u_n} \|s_p^{(2)}(t)\|_2^2 dt \quad (26)$$

for a specified  $\lambda \in [0, \infty)$  and a sequence of knots  $\{u_i\}_1^n$ . Another similar criterion is due to Dierckx [Die95]. Here the smoothness is enforced across the knots, by

minimizing the sum of discontinuity jumps in the  $k^{\text{th}}$  derivative at the knots  $u_i$ ,

$$\lambda \sum_{i=1}^n w_i \|\mathbf{p}_i - s_p(v_i)\|_2^2 + \sum_{i=1}^m \|s_p^{(k)}(u_{i+}) - s_p^{(k)}(u_{i-})\|_2^2. \quad (27)$$

Contrary to the natural smoothing spline criterion the number of knots  $m$  is independent of the number of sample points and corresponding parameter values  $v_i$ .

The smoothing method based on springs (closed case) searches for a discrete set of points  $\{\mathbf{x}_i\}_0^{n-1}$  which minimize

$$\lambda \sum_{i=0}^{n-1} w_i \|\mathbf{p}_i - \mathbf{x}_i\|_2^2 + \sum_{k=1}^s \sum_{i=0}^{n-1} \left( \kappa_{i-1} |x_i^{(k)} - x_{i-1}^{(k)}| + \kappa_i |x_i^{(k)} - x_{i+1}^{(k)}| \right)^2, \quad (28)$$

where the indices are incremented and decremented modulo  $n$ . All these criteria can thus be written in the form

$$\lambda \delta + \eta, \quad (29)$$

where  $\delta$  represents the deviation and  $\eta$  the smoothness terms. In our case, the smoothness term arises from the convex combination requirement for each point, and is related to the total potential spring energy of the system in the sense that it tries to proportionally preserve the distances between the points. In case of a non-closed curve, when there is no restriction on position of the interior points, the method will yield a minimal energy configuration with all points lying on one line. Compared with the smoothing splines, the main difference of the SSE-smoothing method is its discrete nature. In particular there is no underlying continuous function or any knots or parameter values.

## 4 Smoothing with max-norm constraints

In some applications it is required that the smoothed curve should lie within some predefined bounds w.r.t. the original points. This is especially useful in reverse engineering and quality control applications, where it is important for the smoothed data not to deviate too much from the measured data in *any* given point. The criterion from previous section keeps the weighted *average* quadratic distance to the points less than some threshold  $\tau/n$ . A more natural requirement would be to limit the *maximal* deviation of the smoothed points. This is achieved by redefining the deviation constraint in max-norm terms. The smoothing problem formulation from section 3 becomes

$$\text{minimize } \sum_{k=1..s} \|\mathbf{A}\mathbf{x}^{(k)}\|_2^2, \text{ s.t. } \max_{i=1..n} \omega_i \|\mathbf{p}_i - \mathbf{x}_i\|_\infty \leq \tau_{max}. \quad (30)$$

In each dimension  $k$  the problem can be formulated as a *quadratic programming* optimization problem with box constraints, i.e.

$$\begin{aligned} & \text{minimize } \mathbf{x}^{(k)\top} \mathbf{A}^\top \mathbf{A} \mathbf{x}^{(k)} \\ & \text{s.t. } \frac{-\tau_{max} + \mathbf{d}_i^{(k)}}{\omega_i} \leq x_i^{(k)} \leq \frac{\tau_{max} + \mathbf{d}_i^{(k)}}{\omega_i}. \end{aligned} \quad (31)$$

The matrix  $\mathbf{A}^\top \mathbf{A}$  is symmetric and positive semi-definite, therefore the solution to this system is global, but not necessarily unique. Various algorithms exist for solving this kind of problems. In Matlab's Optimization Toolbox a particularly fast algorithm is provided under the name `linlsq`. It is a trust-region reflective Newton method specially adapted to handle large sparse, bounded least-squares problems. During the optimization a number of conjugate gradient iterations are applied to systems  $M_k \mathbf{s}_k = \mathbf{g}_k$  (Newton step), where  $M_k$  has the same structure as  $\mathbf{A}^\top \mathbf{A}$ . Since  $\mathbf{A}$  is tridiagonal and  $\mathbf{A}^\top \mathbf{A}$  is symmetric with bandwidth equal to five, the conjugate gradient method in combination with a preconditioner with bandwidth five gives the best results. In Matlab the default preconditioner of this kind is based on Cholesky factorization of some band approximation matrix of  $M_k$ .

To solve the systems  $M_k \mathbf{s}_k = \mathbf{g}_k$  needed for the Newton step, it is possible to use a sparse direct QR solver as well. In this case the solution discussed in section 3.3 is applicable to this optimization problem. In our experiments, the Matlab's sparse QR solver tended to be slightly slower than the preconditioned conjugate gradient approach.

Obviously the  $\tau_{max}$  parameter can be specified for each point. In this way it is possible to obtain interpolation at some points. This might be useful for non-closed curve smoothing if we want interpolation in the end points.

## 5 Experimental results

### 5.1 SSE-smoothing

Figure 6(a) shows the Fermat's spiral with some superimposed random noise. This is an example of an open curve with anisotropic point density, therefore we used ball neighborhoods of constant radius  $R = 0.5$ . In 6(b) a polygonal reconstruction is shown of the noisy sample points. In subsequent figures 6(c)-6(e) the effect of smoothing with unit weights is shown. In the first figure we have a slight overfit, the third deviates too much from its original curve, while the second resembles the original spiral very well.

Figure 7(a) shows a point set obtained by sampling a lizard in one of Escher's graphical works. The reconstruction, see figure 7(b), was computed using the

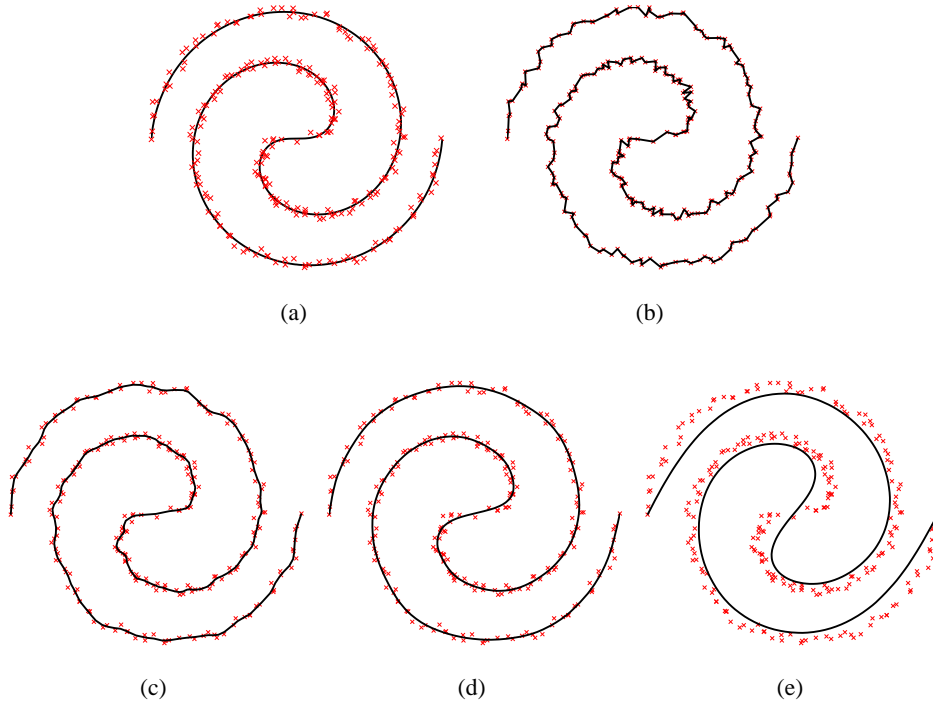


Figure 6: (a) Fermat's spiral with polar equation  $r^2 = \theta$  with noise (b) polygonal reconstruction of the noisy spiral (c) SSE-smoothing with  $\tau = .5$  (d)  $\tau = 1$  (e)  $\tau = 10$ .

meshless parameterization method in the plane with 5 nearest neighbors. Figure 7(c) shows the smoothing spline and figure 7(d) shows the smoothed polygon computed by the springs method. The error of both is comparable in the sense that if measured at the parameter values for the smoothing spline and at the polygon vertices for the SSE-smoothing method, the error is the same. As can be seen, the two figures are quite similar. However in general the SSE method suffers from the 'shrinkage' effect as the 'springs' tend to collapse when there is less restriction on the deviation term in (28).

## 5.2 Weighting functions

In figure 8(a) some possible weight functions  $\psi(r)$  are shown. We distinguish the families of scaled functions defined on  $(0, 1]$ :  $1/r^k$ ,  $e^{-k}$ ,  $e^{-k^2}$ ,  $(1-r)^k$  and  $\psi(r) = 1$ . For the reconstruction step it is best to use rapidly decaying functions.

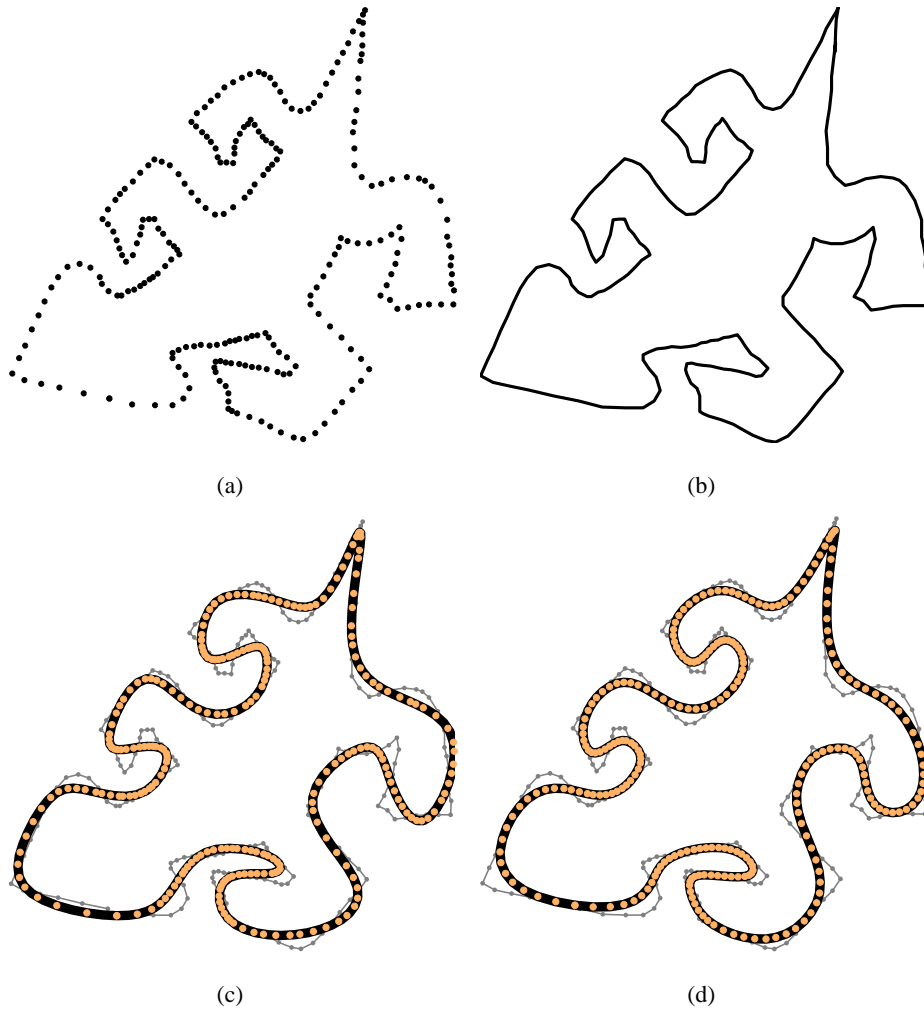


Figure 7: (a) points sampled from the Escher's lizard in his wood engraving work 'Smaller and Smaller', 1956. (b) polygonal reconstruction by the meshless parameterization method. (c) smoothing of the polygon by the smoothing spline of degree 3, with 40 knots and the smoothing criterion of Dierckx (d) result of SSE-smoothing.

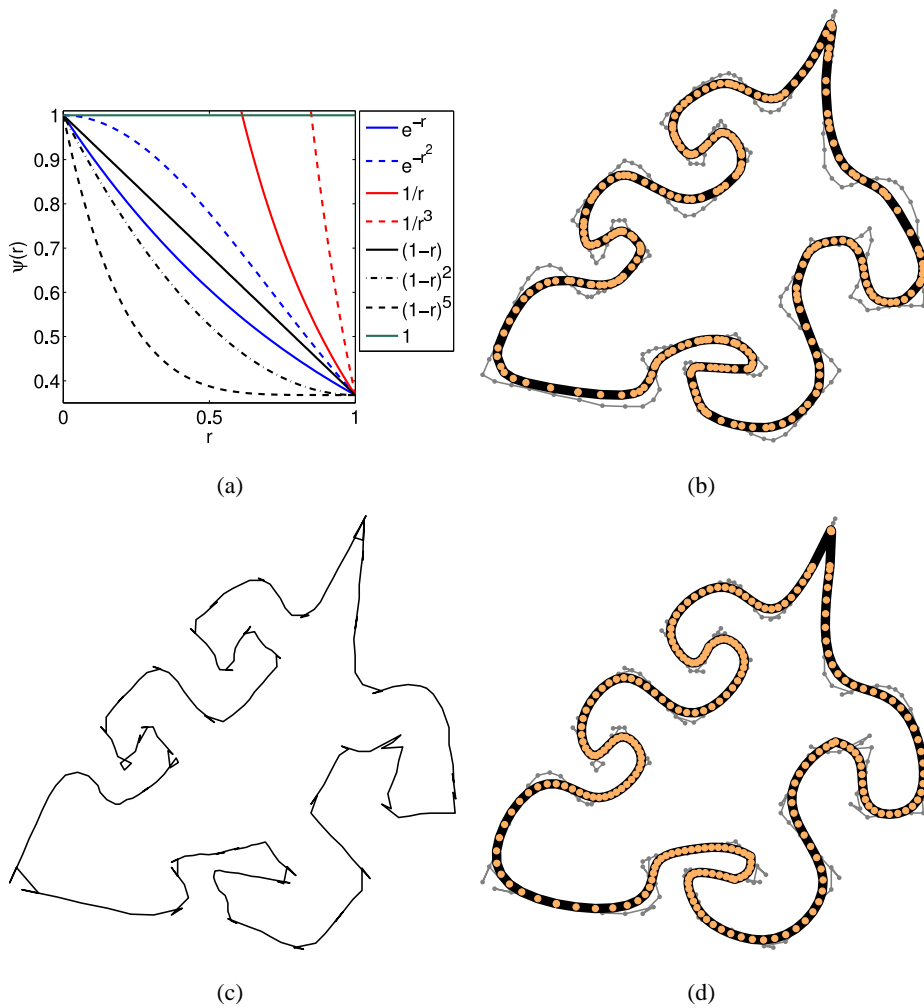


Figure 8: (a) weight functions  $\psi(r)$  (b) SSE-smoothing with  $\psi(r) = 1/r^2$  (c) reconstruction with  $\psi(r) = 1$  (d) SSE-smoothing with  $\psi(r) = 1$ .

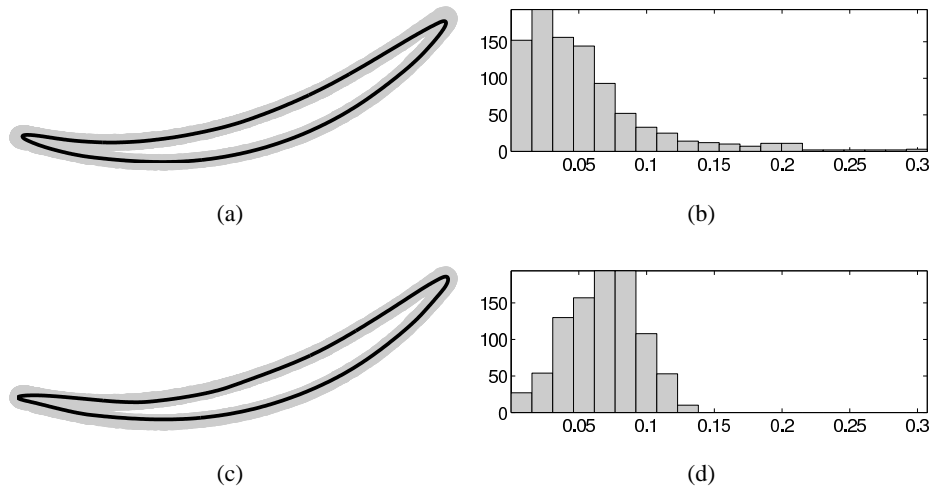


Figure 9: Turbine blade section example: (a) the result of SSE-smoothing with  $\tau = 5$  (b) the histogram shows the number of points w.r.t. the Euclidean distance to their original positions (c) max-norm smoothing with  $\tau_{max} = 0.0933$  (d) same as (b) but for the max-norm case as shown in (c).

For smoothing the functions of type  $1/r^k$  seem to be less appropriate, because the nearby points seem to cluster, yielding a less smoother looking polygon, see figure 8(b). Therefore, for the combined application of reconstruction and smoothing, the most appropriate weight functions are  $e^{-k}$ ,  $(1-r)^k$ . The smoothing process is also quite insensitive to small reconstruction errors. As an example in figure 8(c) a constant weight function was used, which is obviously not suited for reconstruction and does not produce a valid reconstruction. However, when smoothing is applied to this polygon with the same weights, the result does not differ significantly from examples obtained from a correct reconstruction and decaying weights as in figures 7(d) and 8(b). Note that in this case the matrix  $A$  is the usual (Tutte) Laplacian matrix of the reconstructed polygon (seen as a graph), making our smoothness criterion the same as in [SCO04].

### 5.3 Max-norm smoothing

In figure 9 a cross section of a turbine blade is shown. This section was obtained by a laser scanner measurement with the resulting non-uniform, noisy point data consisting of about 1000 points. The gray area in figures 9(a) and 9(c) represent some offset region of the measured points. The black line is the smoothed polygon.

model	points	reconstruction	SSE-smoothing	max-norm smoothing
lizard	250	0.40s	0.09s (5 iter)	0.50s
spiral	251	0.24s	0.11s (6 iter)	0.44s
blade section	927	3.88s	0.14s (5 iter)	1.81s

Figure 10: Indicative timings for the SSE and max-norm methods

When applying the SSE-smoothing procedure the approximation suffers from the well known shrinkage effect near the two endpoints of the blade. While the deviation in most of the points is small, the smoothed samples on the endpoints produce large distances as can be seen on figure 9(b). If we want the maximal deviation to be limited (by  $\tau_{max}$ ) we can use the max-norm smoothing. To obtain comparable results we use  $\tau_{max} = 0.0932$ , such that the average quadratic deviations of 9(a) and 9(c) are approximately equal. The result is shown on figure 9(c) and the corresponding distribution of distances in 9(d). The maximal deviation requirement comes at a price, however. The curve in 9(c) is less smooth, in the sense that the residual  $\sum_{k=1..s} \|A\mathbf{x}^{(k)}\|_2^2$  is greater than in 9(a).

## 5.4 Timings

In figure 10 we give some indicative timings for the examples in this paper, measured on a P4 2.4Ghz machine. All the algorithms were implemented in Matlab, and all solvers used are the sparse solvers provided by Matlab. The reconstruction column contains the overall time needed for the reconstruction, including the computation of the matrix  $W$ .

## 6 Conclusion

The contribution of this paper is twofold. First we presented an extension of the polygonal curve reconstruction method of Floater, to closed, non-selfintersecting curves. Compared to other reconstruction techniques this method requires only the solution of some sparse linear systems. It works for points in  $\mathbb{R}^s$  and always yields a polygonal curve connecting *all* the given sample points.

Furthermore we showed that it is possible to perform discrete smoothing of the reconstructed curve in a manner comparable to that of smoothing splines. The smoothing can be performed under two different constraints: the SSE constraint and the max-norm constraint. It is based on the *convex combination* formulation, also used in the reconstruction step. Therefore the smoothing algorithm can reuse the information computed in the reconstruction step. One of the advantages of the

smoothing algorithm is that it requires only a (sparse) least squares solver. Due to the special structure of the involved matrices the solver can be implemented to run in *linear* time and memory.

We should also mention the very recent work of Sorkine et al. [SCOIT05] on the so-called geometry-aware bases, which provides a more general discussion of the least-squares meshes [SCO04]. Our smoothing method can be viewed as a variant of the curve reconstruction method using geometry-aware bases. However, in our approach we emphasize the *constrained smoothing* aspect instead of the *progressive* approach. Compared to [SCOIT05] our approach uses all sample points as anchor points at the same time, while satisfying some constraints on the deviation of the smoothed points. Furthermore Sorkine et al. uses a Laplacian matrix as the smoothing term, while the matrix  $A$  used in our least-squares system contains non-equal weights and has a very special structure, because for each point there are exactly two neighbors. This makes it possible to solve the system in a very efficient way.

In the future we plan to investigate the applicability of constrained smoothing to more general geometries such as meshes and point clouds.

## 7 Acknowledgements

This research presents results of the Project IUAP P5/22 funded by the Interuniversity Attraction Poles Programme - Belgian Science Policy. The scientific responsibility rests with the authors. The ‘turbine blade’ point cloud is courtesy of Metris N.V. Belgium.

## References

- [ABE98] AMENTA N., BERN M., EPPSTEIN D.: The crust and the  $\beta$ -skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing* 60 (1998), 125–135.
- [AM00] ALTHAUS E., MEHLHORN K.: TSP-based curve reconstruction in polynomial time. In *Proc. of the eleventh annual ACM-SIAM symposium on Discrete algorithms* (2000), SIAM, pp. 686–695.
- [BB97] BERNARDINI F., BAJAJ C.: Sampling and reconstructing manifolds using alpha-shapes. In *Proc. of the Ninth Canadian Conference on Computational Geometry* (1997), pp. 193–198.

- [CBM\*03] CARR J. C., BEATSON R. K., MCCALLUM B. C., FRIGHT W. R., MCLENNAN T. J., MITCHELL T. J.: Smooth surface reconstruction from noisy range data. In *ACM GRAPHITE 2003* (2003), ACM Press, pp. 119–126.
- [Cox81] COX M. G.: The least squares solution of overdetermined linear equations having band or augmented band structure. *IMA Journal of Numerical Analysis* 1 (1981), 3–22.
- [dB78] DE BOOR C.: *A practical guide to splines*. Springer-Verlag, 1978.
- [dG95] DE FIGUEIREDO L. H., GOMES J.: Computational morphology of curves. *The Visual Computer* 11, 2 (1995), 105–112.
- [Die95] DIERCKX P.: *Curve and Surface Fitting with Splines*. Oxford University Press, 1995.
- [DK99] DEY T. K., KUMAR P.: A simple provable algorithm for curve reconstruction. In *Proc. 10th. ACM-SIAM Symposium on Discrete Algorithms (SODA '99)* (1999), pp. 893–894.
- [DMR99] DEY T. K., MEHLHORN K., RAMOS E. A.: Curve reconstruction: Connecting dots with good reason. In *Proc. 15th annual ACM Sympos. Comput. Geom.* (1999), pp. 197–206.
- [Ede98] EDELSBRUNNER H.: Shape reconstruction with delaunay complex. In *Proc. Sympos. Latin American Theoret. Inform.* (1998), Springer-Verlag Lecture Notes 1380, pp. 119–132.
- [EKS83] EDELSBRUNNER H., KIRKPATRICK D. G., SEIDEL R.: On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory* 29, 4 (1983), 551–559.
- [Flo03] FLOATER M.: Analysis of curve reconstruction by meshless parameterization. *Numerical Algorithms* 32 (2003), 87–98.
- [FR01] FLOATER M., REIMERS M.: Meshless parameterization and surface reconstruction. *Comp. Aided Geom. Design* 18 (2001), 77–92.
- [Gie99] GIESEN J.: Curve reconstruction, the travelling salesman problem and menger’s theorem on length. In *Proc. of the Fifteenth Annual Symposium on Computational Geometry* (June 1999), ACM Press.
- [GL96] GOLUB G. H., LOAN C. F. V.: *Matrix Computations*. John Hopkins University Press, 1996. Third Edition.

- [Lee00] LEE I.-K.: Curve reconstruction from unorganized points. *Comp. Aided Geom. Design* 17 (2000), 161–177.
- [SCO04] SORKINE O., COHEN-OR D.: Least-squares meshes. In *Proceedings of Shape Modeling International* (2004), IEEE Computer Society Press, pp. 191–199.
- [SCOIT05] SORKINE O., COHEN-OR D., IRONY D., TOLEDO S.: Geometry-aware bases for shape approximation. *IEEE Transactions on Visualization and Computer Graphics* 11, 2 (2005), 171–180.
- [Sed02] SEDGEWICK R.: *Algorithms in C : Part 5, Graph Algorithms*, 3d ed. ed. Addison-Wesley, 2002.
- [Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, pp. 351–358.
- [VRV03] VOLODINE T., ROOSE D., VANDERSTRAETEN D.: Efficient triangulation of point clouds using Floater parameterization. In *Proc. of the Eighth SIAM Conference on Geometric Design and Computing* (2003), Nashboro Press, pp. 523–536.