Simple Reconstruction of Non-Simple Curves

Tobias Lenz
tlenz@mi.fu-berlin.de

Institut für Informatik
Freie Universität Berlin
Takustr. 9
14195 Berlin, Germany

Technical Report B 05-02

Abstract

The presented algorithm reconstructs collections of *arbitrary* curves (open, closed, smooth, with corners, with or without intersections). The algorithm is very simple and short and has a novel and very simple sampling condition which guarantees correct results and does not need special adaption for regions close to corners, endpoints or intersections. The corner and intersection points are not required to but allowed to be in the sample. The described method works for curves in any dimension d asymptotically in $\mathcal{O}(n^{2-1/d})$ time with involved data structures. Experiments show already a good performance with a very simple kd-tree structure.

1 Introduction

1.1 What is Curve Reconstruction

Answering the question up-front with the words of Tamal Dey, Kurt Mehlhorn and Edgar Ramos: Curve reconstruction is "connecting dots with good reason" [6]. The "dots" are so called *samples* taken from the curve—whatever that may be precisely—one wants to reconstruct. The outcome should also be a curve approximating the original curve the samples were taken from. Typically one creates a piecewise linear approximation of the original curve. So it is in fact "connecting the dots" with the goal to find the order which satisfies ones needs best.

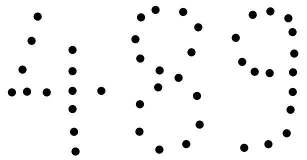


Figure 1: How to connect these dots?

Regarding it in a more abstract fashion, in curve reconstruction only a subset of the original information is given and the task is to complete it in the most likely way. This is something very natural, e.g. the human visual system does it all the time. Figure 1 shows only some dots but it should be no problem to identify them as the number 489. Although the dots are not connected, they somehow indicate the shape of this number and the human brain automatically reconstructs the missing information.

This is exactly the task the computer should solve for us. The results of course depend on the positions of the given points and the used algorithm to connect them. This paper is mainly concerned with algorithms providing provable results, such that for an input satisfying a certain *sampling condition*, the corresponding algorithm computes an output with certain guarantees with respect to the original curve.

Although the reconstruction of two-dimensional objects (curves) has applications on its own, it mainly provides foundations and new ideas for *surface reconstruction* in three dimensions.

1.2 Sampling Conditions

The quality of the reconstructed curve depends heavily on the quality and quantity of the input points. The following sections bring up some ideas of curve reconstruction algorithms. They guarantee a “correct” reconstruction if the samples satisfy certain conditions.

All of the discussed sampling conditions somehow limit the allowed positioning of sample points not only locally, regarding a single point and maybe its neighbors, but globally, enforcing properties of the set of points as a whole, e.g. utilizing the *medial axis*. Most of them are also not easy to verify, but in general—uniform sampling excluded—they all lead to the following rule of thumb: parts of the curve which are close to other parts or with high curvature should be sampled more densely than straight and distant parts. This sounds very natural and is intuitively a good idea.

Many other sampling restrictions are imaginable, like the one in this paper. In general a special sampling condition suits best for a special algorithm, providing a sufficient condition for a correct reconstruction.

1.3 Filtering Delaunay Edges

Reconstructing a curve from n given sample points means picking a subset of the edges of the embedded complete graph with these n vertices. However this is done, there are $\Theta(n^2)$ edges possible to choose. If an algorithm only handles simple curves (without intersections) the reconstruction is a planar graph which has $\mathcal{O}(n)$ edges. This allows to pre-select a linear number of edges from which the edges for the final reconstruction are chosen. Due to some for curve reconstruction extremely appropriate properties, the edges of the Delaunay triangulation are often taken. They have a nice relative positioning, can be computed efficiently and the concept naturally extends to higher dimensions, reconstructing surfaces.

According to these advantages it seems obvious that numerous algorithms use the Delaunay triangulation as their edge basis and then throw out edges step by step until the remaining edges form the final reconstruction. Many of these algorithms require the sample to be a so called *ε-sample* fulfilling the condition that for each point on the curve the distance to the closest sample point is at most ε times the *local feature size*. The local feature size of a point is the distance to its closest point on the *medial axis* and was introduced by Amenta, Bern and Eppstein [2] to distinguish between smaller features needing a high sampling density and larger features which only need a low sampling density to be captured well. This is explained more elaborately in section 3.1.

One important idea for this work came from Dey and Kumar and their Nearest Neighbor Crust algorithm [5]. They connect each point p with its nearest neighbor n and its nearest half-neighbor h which is basically the nearest neighbor among the points q_i with $\angle \overline{np}, \overline{pq_i} > 90°$. Picking the nearest neighbor essentially is like growing a circle around a point until another point touches the boundary. We do basically the same but instead of growing circles we use other shapes which are introduced in a later section.

1.4 Other Algorithms

The problem of reconstructing closed curves might also be formulated as an instance of the well known traveling salesperson problem as done by Giesen [10, 11]. The sample points are the sites which all must be visited on the shortest tour possible. The very basic idea is if the sample is dense enough such that two points adjacent on the curve are closer to each other than to any other point, the TSP tour will be the correct reconstruction. Later Althaus and Melhorn [1] showed that this tour can be found in polynomial time.

Other approaches use additional information, e.g. normals for the sample points like the so called tensor voting.

The surface reconstruction by Cohen-Steiner and Da [4] uses a greedy technique growing the reconstruction adding the most fitting triangles one at a time which is roughly similar to the method proposed in this paper for curves.

1.5 Reconstructible Curves

All the numerated algorithms have one thing in common. They cannot reconstruct arbitrary curves, applying a very general definition of the term curve. The very first algorithms worked for simple, closed, smooth curves only [2]. Newer algorithms work for open curves [6], curves with corners [7] or collections of curves [9]. Some of them are for 1-manifolds, some work well with T-junctions but the big challenge of reconstructing self-intersecting curves or several overlapping curves is still present. One reason for that is the wide use of Delaunay triangulations which is asymptotically fast but naturally excludes intersections if one is not willing to handle them with special cases.

This problem is tried to tackle in this paper; reconstructing a collection of very general curves, each may be open or closed, with a finite number of corners, intersections and self-intersections.

Section 2 provides the necessary definition of curves and their properties. In section 3 the problems with certain features like intersections are explained and the main ideas to solve them are presented. The next two sections contain the algorithm and the required sampling condition. The correctness and the analysis of time and space consumption is done in the sections 6–8. The paper concludes with implementation details and a comparison to ε-sampling based techniques.

2 Curves and Their Variants

2.1 Properties of Curves

Please note, that the following definitions are chosen with respect to useful terminology within this paper and may differ from definitions found elsewhere.

Definition 2.1. Let σ be a continuous and piecewise injective mapping $\sigma : [0; 1] \to \mathbb{R}^d$ for a fixed d. We call the image of σ a (d-dimensional) *curve*[1].

We will loosely speak of *the curve* σ which always refers to the image of σ. In general we will denote the curve generated by the mapping σ or σ_i by Σ or Σ_i respectively.

Definition 2.2. A finite *set of curves* (or *collection of curves* or just *collection* for short) generated by m mappings σ_i for $1 \le i \le m$ is the union of the m curves: $\Sigma = \bigcup_{i=1}^{m} \Sigma_i$.

[1] It is appropriate to treat closed curves as mappings from $\mathbb{S}^1$ to $\mathbb{R}^d$ to omit problems and special cases for the interval boundaries.

Since a single curve can also be regarded as a collection generated by a single mapping, we will only speak about reconstruction of collections from now on, which completely includes reconstruction of single curves.

Please note that the mapping generating a certain curve is not unique, neither is the set of mappings generating a collection. Different mappings leading to the same collection are called *reparameterization* of each other.

Definition 2.3. A curve is called *closed* if $\sigma(0) = \sigma(1)$[1]. Otherwise it is called *open* and $\sigma(0)$ and $\sigma(1)$ are called *endpoints*.

Definition 2.4. A point p on a curve is called *self-intersection* if and only if $a, b \in [0; 1]$ exist such that $a \neq b$ and $\{a, b\} \neq \{0, 1\}$ and $\sigma(a) = \sigma(b) = p$.

Definition 2.5. A curve is called *simple* if and only if it contains no self-intersections.

The simple curves are exactly the 1-manifolds. Open simple curves correspond to 1-manifolds with boundary (the endpoints) while closed simple curves are 1-manifolds without boundary.

Definition 2.6. A point $\sigma(c), c \in [0; 1]$ on a curve is called *corner* if and only if $\sigma(c)$ is not an endpoint and there is no (unique) tangent in $\sigma(c)$.

Definition 2.7. A curve is called *smooth* if and only if it contains no corners and no self-intersections.

Definition 2.8. A point p in the collection of the curves $\Sigma_1, \ldots, \Sigma_m$ is called *intersection* if and only if either p is a self-intersection of any Σ_i or there exist $\Sigma_i \neq \Sigma_j, 1 \le i, j \le m$ such that $p \in \Sigma_i \cap \Sigma_j$.

Figure 2 gives an intuition of some of the defined features i.e. smooth, open and closed parts, endpoints, corners and intersections.

The natural expectation of a *curve* seems to include certain nice properties like continuity and at least piecewise differentiability. Some of these restrictions are in fact necessary to get a reasonable reconstruction and thus are common among reconstruction algorithms. For this paper the restrictions to *reconstructible collections* are stated as follows.

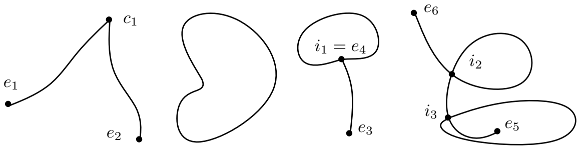


Figure 2: Collection example with endpoints e_j, corners c_j and intersections i_j

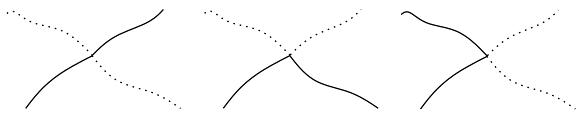


Figure 3: Two curves (solid and dotted) generating a junction in three different ways although the collection always looks the same. Is it a normal intersection (left) or two corners touching each other (center and right)?

Definition 2.9. A collection of curves is said to be *reconstructible* if and only if it has finitely many corners and intersections.

No algorithm is known which handles at least partly curves which are not reconstructible due to this definition.

From now on speaking of collections we will always mean reconstructible collections if not explicitly stated differently.

2.2 Picking Samples from a Curve

In curve reconstruction the task is to find an approximation of the original curve based on a preferably small set of points on that curve. This set is called *sample* and defined as follows.

Definition 2.10. A *sample* S is a finite subset of a collection Σ. The elements of the sample are points in $\mathbb{R}^d$ and are called *sample points* or *samples* for short.

Intending to apply a special algorithm, the sample points should fulfill some conditions to guarantee a reconstruction with a certain quality. This is explained in detail in chapter 5.

2.3 Reconstructed Curves

Obviously it is impossible to reconstruct the original mappings or the parameterizations although some ambiguities are already excluded by the definition 2.9.

Basically the reconstructed "curve" is an embedded graph which has the sample points as vertices and the edges are computed by the used reconstruction algorithm. We will call this graph *reconstruction* and denote it by Γ. If a piecewise linear approximation is not suitable for a specific application, existing techniques to find a smooth approximation based on Γ can be applied [6].

Definition 2.11. Consider a set of points $A \subset \Sigma$. Two points $p, q \in A$ are called *adjacent in* Σ *restricted to* A if and only if a continuous mapping $\varphi : [0; 1] \to \Sigma$ exists such that there are $a, b \in [0; 1] : \varphi(a) = p, \varphi(b) = q$ (without loss of generality $a < b$), and $\forall c \in (a; b) : \varphi(c) \notin A$.

In other words, two points p, q are adjacent in A if and only if a path from p to q along the collection exists which does not contain any points from $A \setminus \{p, q\}$.

Definition 2.12. Let S be a sample of the collection $\Sigma = \bigcup_{i=1}^{m} \Sigma_i$ generated by the m mappings $\sigma_i, 1 \leq i \leq m$. An edge between $p, q \in S$ in a reconstruction of Σ is a *correct edge* if and only if an index $1 \leq k \leq m$ exists such that p and q are adjacent in Σ_k restricted to S.

3 Non-smooth Features

3.1 The Intersection Challenge

The ε-sampling is very well tailored for the use of Delaunay triangulations and for smooth curves the ε-sampling together with the *medial axis* and the *local feature size* [2] are very expressive objects respectively quantities. Unfortunately some major drawbacks emerge for reconstructing arbitrary curves.

The ε-sampling is an intuitive condition which basically states that the sampling should be dense if the curvature is high or several parts of the curve are close together while it might be loose in straight regions of the curve. Formally it is defined as follows.

Definition 3.1. The *medial axis* of a smooth curve is the set of points M containing the center points of all empty circles which touch the curve in more than one point.

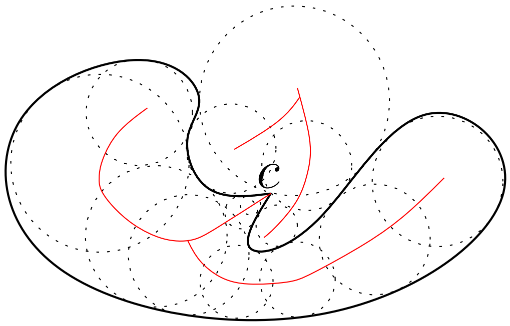


Figure 4: A non-smooth curve (black) with some medial balls (dotted) and the medial axis (red) intersecting the curve at the corner point c.

Definition 3.2. The *local feature size* of a point $p \in \Sigma$ is defined as $\mathrm{lfs}(p) = \min_{m \in M} \|p - m\|$.

Definition 3.3. A sample S is an *ε-sample* of the smooth curve Σ if and only if $\forall p \in \Sigma : \exists s \in S : \|p - s\| \leq \varepsilon \cdot \mathrm{lfs}(p)$.

These definitions apply well to smooth curves but an ε-sample is plainly not possible for curves with corners or intersections because the medial axis goes through these points and hence the local feature size becomes zero. Therefore the sampling density should be infinite. See figure 4 for an illustration. A possible solution is to exclude a small neighborhood of these points from the ε-sampling condition and define a new one for these regions. This is used for corners in [9]. An extension to the Crust algorithm with a special condition for regions around intersections was recently introduced by Knobelsdorf [12].

Corners and endpoints can be handled by a change in density. This is very natural and should be clear from figure 5. Unfortunately such a trick is not possible for intersections.

Comparing the Delaunay triangulation of a sample with the correct reconstruction one will find some wrong edges in the vicinity of intersections. They have interesting characteristics which are explained and exploited in the next section.

3.2 Key Idea: Inflating Probes

An observations for smooth curves or smooth parts of curves near intersection points is that the angle between segments of properly placed sample points is small while mostly this angle abruptly increases connecting to a wrong point, see figure 6. Therefore a key idea is to favor small angles over small distances.

Of course this must not be exaggerated. Only considering the angle is prohibitive because the algorithm would always connect groups of three colinear points disregarding their distance.

A nice middle course between angle and distance is achieved by putting a specially shaped *probe* at the end of a correct edge, aligned in the edge's direction and inflating this probe (increasing its size) until it hits a point. By this technique it is possible to "tunnel through" a narrow set of wrong points

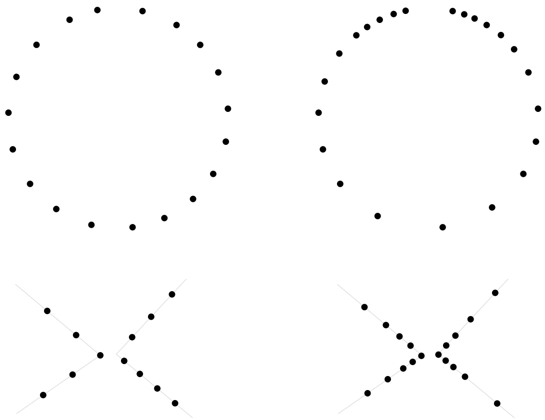


Figure 5: Higher density perceptually indicates disconnectedness. The samples on the left show nearly equally distributed sample points while the samples on the right use higher density close the endpoints and corners to emphasize the upcoming gap.

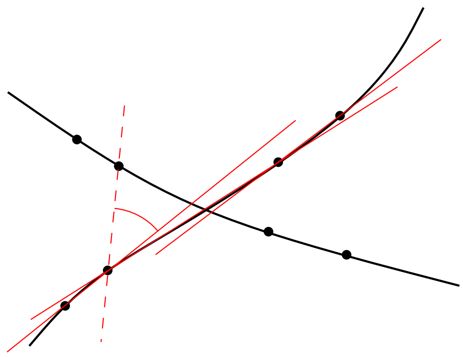


Figure 6: Wrong connections usually create larger angles

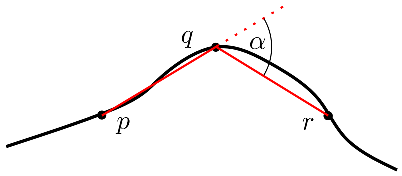


Figure 7: A chain of three sample points and their angle.

and reach the correct one.

To favor small angles over small distances one might use a distance function F which depends exponentially on the given angle parameter but only linearly on the distance parameter. Consider

$$F(d, \alpha) = d \cdot {c_{\text{base}}}^{|\alpha|^{c_{\text{exp}}}}$$

where $d \geq 0$ is in distance units and α is an angle from $[-\pi; \pi]$. We call F our *distance function.* For ease of use a point parameter form is provided as

$$F_{pq}(r) = |q - r| \cdot {c_{\text{base}}}^{|\angle \overline{pq}, \overline{qr}|^{c_{\text{exp}}}}$$

for three points p, q, r. The term *distance function* is precisely defined in definition 3.7 in a later section.

Please note that $F_{pq}(r)$ is finite and strictly positive for every three points p, q, r with $q \neq p$ and $q \neq r$ but not necessarily $p \neq r$. See figure 7 for an illustration of the angle $\alpha = \angle \overline{pq}, \overline{qr}$. Further it is important that F_{pq} is directed and in general different from F_{qp}.

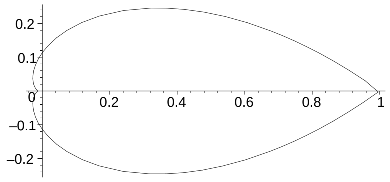
The values for c_{base} and c_{exp} have to be chosen carefully. Figure 8 provides a graphic understanding of these parameters by some examples.

The shapes drawn in figure 8 are probes. In general such a *probe* at the origin in x-direction is represented at best by the following function depending on the angle

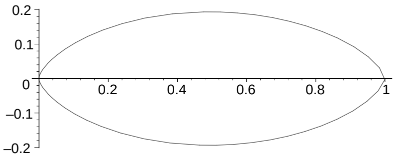
$$\theta(\alpha) = \frac{1}{{c_{\text{base}}}^{|\alpha|^{c_{\text{exp}}}}}, -\pi < \alpha \leq \pi$$

or in terms of the distance function F by the implicit function $\dfrac{1}{F(d, \alpha)} = 1$.

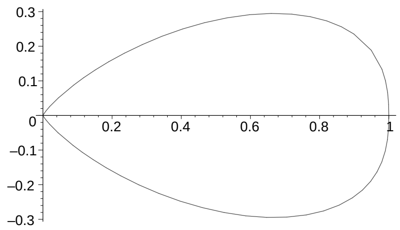
For $c_{\text{base}} = 1$ the probe "degenerates" to a circle which lets the selection criterion behave like a usual euclidean nearest neighbor function.



(a) $c_{\text{base}} = 4, c_{\text{exp}} = 1.0$



(b) $c_{\text{base}} = 16, c_{\text{exp}} = 1.5$



(c) $c_{\text{base}} = 80, c_{\text{exp}} = 3.0$

Figure 8: Polar plots of $\frac{1}{F(d,\alpha)} = 1$ with different parameters c_{base} and c_{exp}

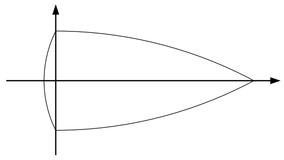
3.3 Convex and Polygonal Probes

The probes described in the previous section have several nice properties. The function F is easily computable for three given points. Moreover the function reflects the intuition to penalize large angles and still favor smaller distances for similar angles.

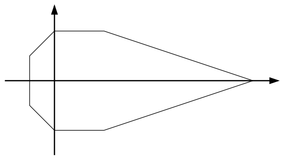
On the other hand there is one major drawback for efficient implementation. The probe is neither convex nor is it easy to compute derivatives or other handy properties. This prohibits most known data structures for efficient queries when it comes to minimizing F_{pq}.

The selection of F in the previous section is not mandatory. Here is a more general definition for a probe which maintains the main property but additionally allows polygonal probes with finite complexity and convex probes.

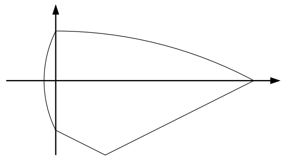
Definition 3.4. A continuous map $\theta : [-\pi; \pi] \rightarrow \mathbb{R}_0^+$ is called *probe* if and only if $\alpha > \beta \geq 0 \Rightarrow \theta(\alpha) < \theta(\beta)$ and $\alpha < \beta \leq 0 \Rightarrow \theta(\alpha) < \theta(\beta)$ and $\theta(-\pi) = \theta(\pi)$.



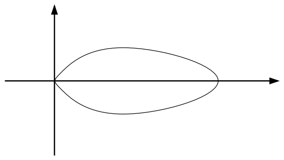
(a) convex, symmetric with negative extend



(b) polygonal, convex, symmetric with negative extend



(c) convex, asymmetric with negative extend



(d) convex, symmetric and without negative extend

Figure 9: Four probes illustrating the variety of possible shapes. All these probes are convex and can be described with constant complexity. Probe (a) is assembled out of three circular arcs, (b) is a polygon with seven vertices, (c) is a combination of circular arcs and straight segments and (d) is a spline.

The name "probe" refers to the shape one obtains from drawing the function θ in polar coordinates for the full circle between $-\pi$ and π with θ as distance from the origin. Some interesting properties of such a probe are the following.

Definition 3.5. A probe θ is *symmetric* if and only if $\forall \alpha \in (0; \pi] : \theta(\alpha) = \theta(-\alpha)$.

Definition 3.6. A probe θ has *negative extend* if and only if $\exists \alpha : \|\alpha\| > \frac{\pi}{2} \wedge \theta(\alpha) > 0$.

Figure 9 shows shapes which are probes by definition 3.4.

Most useful for general curve reconstruction are symmetric probes. If the curves are not closed and smooth, the probe must have a negative extend to detect endpoints and undersampled sharp bends or corners. To further allow the usage of simple efficient data structures, the probe should be convex and for easy computations also polygonal.

Now it is time to define the distance function depending on the probe.

Definition 3.7. Let θ be a probe by definition 3.4. The *distance function* for three points p, q, r is defined as

$$F_{pq}(r) = \begin{cases} \infty & \text{if } \theta(\alpha) = 0 \\ \frac{\|q-r\|}{\theta(\angle \overline{pq}, \overline{qr})} & \text{if } \theta(\alpha) > 0. \end{cases}$$

This definition is valid for any dimension since one considers only the plane spanned by p, q, r and computes $F_{pq}(r)$ in that plane.

3.4 Curve Tracing

One arising problem of the probe concept is that a probe needs an alignment and therefore needs an already reconstructed edge which provides the proper alignment to find the next correct edge. This leads intuitively to the inductive idea to "walk along the reconstruction" or just tracing the curve starting from a specific *seed edge*.

Compared to the global approach of triangulating a point set and then selecting a subset of the triangulation edges, tracing is a more local concept. Extending the reconstructed graph at its "endpoints" with minimum weight components is also a very natural procedure and intuitively clear. This idea is in line with famous algorithms like Dijkstra's shortest path algorithm or Prim's algorithm to construct a minimum spanning tree.

Inflating probes together with tracing curves already provides a nice straightforward technique to reconstruct non-overlapping collections of smooth curves. Sharp corners might become reconstructed by the same idea, first going straight into the corner and then backwards out to a close point (if the apex of the corner is in the sample). Endpoints are the extreme case of corners where the point one came from is the point with minimum distance, i.e. $F_{pq}(p)$ is minimal.

For arbitrary collections including intersections, additional problems may arise which are covered by the sampling condition in section 5.2.

4 The Algorithm

4.1 A General Framework

In this section a general framework for algorithms using tracing and inflation is presented. Finding seed edges and finding consecutive edges might be controlled by the user by selecting criteria appropriate for the current problem respectively changing line 3 to find seed edges and providing a proper function F like the ones discussed in the prior sections.

Algorithm 1: Framework for the reconstruction of collections of arbitrary curves using the tracing technique.

Input: Sample S fulfilling the sampling condition
Output: Polygonal reconstruction Γ of the curve Σ

begin
1 $\Gamma \leftarrow \emptyset$
2 **while** $|S \setminus \Gamma| \geq 2$ **do**
3 find seed edge (p, q) with $p, q \in S \setminus \Gamma$
4 $\Gamma \leftarrow \Gamma \cup \{(p, q)\}$
5 `processEdge` (p, q)
6 `processEdge` (q, p)
end

Procedure `processEdge`(p,q)

begin
7 find $r \in S \setminus \{q\}$ which minimizes $F_{pq}(r)$
8 **if** $(q, r) \notin \Gamma$ **then**
9 $\Gamma \leftarrow \Gamma \cup \{(q, r)\}$
10 `processEdge` (q, r)
end

The reconstruction algorithm plainly realizes the idea of the inflating probe. Beginning with an *seed edge*—e.g. the shortest edge available—the point minimizing the distance function F as defined in 3.7 is computed and the corresponding edge created. This is continued in both directions until an already existing edge is detected and until all possible seed edges are used up. The reconstructed edges are treated as undirected edges.

No sophisticated structure is used to find the points minimizing F to keep the algorithm simple. It should be blindingly easy to implement this version without the need for any geometric library.

The while-loop reconstructs connected components until less than two unused points are left and thus no additional seed edge exists. In the loop, in line 3, a new seed edge is selected as the closest pair of unused points and added to the reconstruction. The recursion is "seeded" from that edge with both possible orientations.

In the recursive procedure the point r minimizing F with respect to a given edge and orientation (p, q) is found. Only if the edge from the endpoint of the given edge to r is not already part of the reconstruction, it is added to the reconstruction and the recursion continues.

4.2 Handling Corners and Intersections

Surprisingly—using the algorithm from the prior section—the reconstruction task does not become harder after introducing corners and intersections. Exactly the same algorithm can also handle piecewise smooth curves with a finite number of corners and intersections.

Starting at an edge (p, q), endpoints are detected if p, q are so close to each other that F_{pq} is minimized by p. This would create the edge (q, p) which is the same as (p, q) and hence is omitted and this branch of the tracing process stops.

5 Sampling Condition

5.1 Reconstruction Depends on Sample, Not on Original Curve

After the definition of a sample in section 2.2 and the introduction of ε-samples in section 3.1 one should discuss the true requirements for samples. In this discussion we agree with Funke and Ramos [8, 9], and extend their point of view to collections with intersections.

The task is to find a finite sample $S \subset \Sigma$ such that the given reconstruction algorithm can deduce a piecewise linear approximation of Σ from S. The sample is not created with respect to good approximation in terms of a certain minimized quadratic error or a similar quality measure. It has to be just dense enough to allow a *correct* reconstruction by definition 2.12. Actually a sparse sample has a big advantage due to shorter running time and it sounds reasonable that reconstructing a sensible curve from few points is a much bigger achievement than doing it from a very detailed sample.

Figure 10 shows an example of an extremely sparse sample which only contains four points. Obviously the approximation error—measured e.g. as symmetric Hausdorff distance—between Γ and Σ

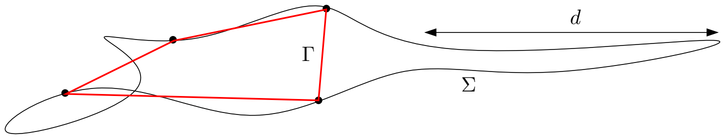


Figure 10: The reconstruction Γ from a very sparse sample disregards many features from the original curve Σ.

becomes arbitrarily large by enlarging d. Some thin features are omitted and even the self-intersection, which might be an interesting artifact, is not represented in the reconstruction. Nevertheless Γ is the best reconstruction one can obtain from the four sample points and our opinion is that it is better to have a reconstruction reflecting the quality of the sample than to have no reconstruction at all. This might also be used deliberately while sampling a curve to focus some regions and neglect others regardless of their real features.

The conclusion from this section is the fact that our sampling condition—opposed to the ε-sampling condition—will not depend on direct geometric properties of the original collection Σ like curvature or *local feature size* but on correlations of points in S only.

5.2 The Sampling Condition

The basic sampling condition is very simple and the idea is comparable with the one given in [8]. However the condition stated here is inductive, having a correct edge it defines another correct one. To get an anchor for this induction another condition is necessary and therefore the following formal decomposition is introduced.

Definition 5.1. A point $s \in S$ which is an intersection in the collection Σ by definition 2.8 ($\Leftrightarrow s$ has a degree larger than 2 in the reconstruction) is called *cut vertex*.

Definition 5.2. Removing the cut vertices from Σ would decompose it into 1-manifolds with or without boundary and "overlapping 1-manifolds" which have common points in Σ but not in S. The resulting set of points is called *1-manifold decomposition* of Σ with respect to S. Formally the cut vertices are not removed and still belong to all adjacent components but the components are separated.

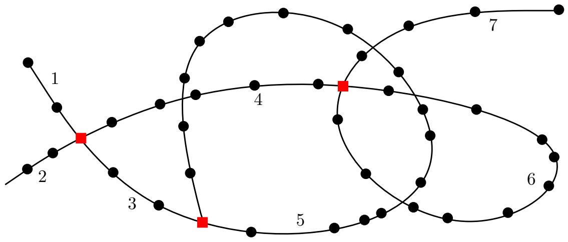


Figure 11: A collection, its sample (dots and squares), cut vertices from the decomposition (squares) and the enumerated components 1–7.

An illustration of this decomposition is given in figure 11, where the dots and squares are the sample points. The points marked by squares are removed during the 1-manifold decomposition, so we end up with seven components.

Utilizing the just defined formal decomposition it is possible to define the sampling condition as follows.

Definition 5.3. An edge (a, b) with $a, b \in S$ is called *seed edge* if and only if the samples a, b are adjacent in the same component C of the 1-manifold decomposition of Σ with respect to S and they uniquely minimize the pairwise distances of samples with at least one part in C, formally

$$\|a - b\| < \min \{\|p - q\| \mid p \in C \wedge q \in S \setminus \{a, b\}\} .$$

Definition 5.4. A sample $S \subset \Sigma$ is called a *valid sample* if and only if the following holds.

1. Every component in the 1-manifold decomposition of Σ with respect to S has at least one seed edge.

2. For every directed pair of samples $p, q \in S$ forming a *correct* edge must exist a point $r \in S \setminus \{q\}$ which uniquely minimizes F_{pq} and (q, r) must be correct. Additionally $r = p$ is only allowed if q is an *endpoint* itself or q is a sample

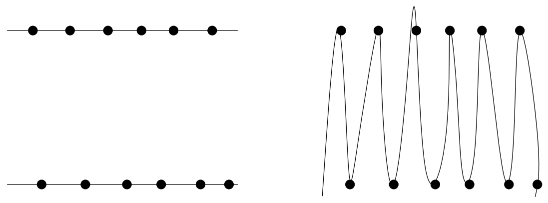


Figure 12: The same sample can be valid for one curve (left) while it is invalid for another curve (right).

adjacent to an endpoint $e \notin S$ in Σ restricted to S.

5.3 Validity Not Verifiable

It would be a desirable function for a reconstruction algorithm to test the sample whether it is valid or not and then tell the user that the results are guaranteed or respectively only heuristics. Unfortunately this is not possible because the definition of a *valid* sample bases on the correct reconstruction and so the correct reconstruction has to be known to check the validity of the sample but this would contradict our task.

Validation of any sample without additional information about the target collection is not possible in general independent of the definition of a valid sample. In the case of ε-samples for example, the sample has to fulfill certain distance conditions based on the original curve and its medial axis. To verify the validity, the original curve has to be given which again renders the task to reconstruct it unnecessary.

Figure 12 shows an example of a valid ε-sample with some fixed ε for two parallel segments on the left which is at the same time an invalid sample of the open smooth curve on the right. If only the sample points are given and we do not know whether they form a valid sample or not, we can not distinguish these cases—independent of the sampling condition.

6 Correctness

6.1 Preparation

A reconstruction from a *valid* sample (see definition 5.4) with n samples can obviously contain at most $\binom{n}{2}$ edges. Only a subset of these edges are correct by definition 2.12.

A reconstruction algorithm is said to be *correct*, if and only if the set of edges generated by the algorithm is exactly the set of all correct edges out of the $\binom{n}{2}$ possible edges. This boils down to the following two implications.

Definition 6.1. A reconstruction algorithm producing a set of edges Γ from a valid sample is *correct* if and only if

1. every edge in Γ is correct by definition 2.12 and
2. every edge which is correct by definition 2.12 is in Γ.

The upcoming two sections will show that the algorithm presented in this paper is correct by the definition given above. The proofs given there will revert to the following short lemma.

Lemma 1. *If the algorithm in section 4.1 processes a correct edge on a component C in the 1-manifold decomposition of Σ, the complete component C will become reconstructed correctly, assuming a valid sample.*

Proof. Every component C in the 1-manifold decomposition is by construction a 1-manifold. Consider two adjacent points $p, q \in C$. In a valid sample (definition 5.4) a correct edge implies correct edges in both orientations.

In the algorithm's calls to `processEdge` it is taken care of that the edges (p, s) and (q, r) incident to the correct edge (p, q) are added to Γ if $r \in S \setminus \{q\}$ minimizes F_{pq} and $s \in S \setminus \{p\}$ minimizes F_{qp} respectively. For a valid sample these edges are correct by part two of definition 5.4 since r, s minimize F_{pq} or F_{qp} respectively.

Since all this happens on a 1-manifold, there are at most two directions available from the edge (p, q) and one can apply the arguments above to the implied edges (q, r) and (p, s) in an inductive manner. $\square$

6.2 All Constructed Edges are Correct

Proof (part 1). We consider a collection Σ, a *valid sample* S of Σ and a reconstruction Γ built from the samples in S. Adjacency in the following paragraphs is always considered with respect to Σ restricted to S.

The algorithm given in section 4.1 obviously starts with the shortest edge (p, q) on some component C of the 1-manifold decomposition. For valid S this edge must be a seed edge and it must exist by part one of definition 5.4. It is correct because p and q have to be adjacent to form a seed edge by definition 5.3.

After line 4 is executed the first time the reconstruction contains only (p, q). Lemma 1 assures that the complete component C including all adjacent cut vertices is reconstructed in the following steps.

It might happen that the algorithm does not stop at a cut vertex and also generates edges on components different from C. Since these edges are implied by correct edges in C they also must be correct by definition 5.4 and again lemma 1 guarantees that these components are reconstructed as a whole. When this process stops, Γ contains at least one component from the 1-manifold decomposition and it only contains complete components and only correct edges.

If no points remain, the algorithm will terminate. Otherwise the shortest edge with endpoints not in Γ is chosen by the algorithm. This is a seed edge and hence correct because no edge of the remaining components from the 1-manifold decomposition is already reconstructed and by part one of definition 5.4 every component has a seed edge and by definition 5.3 only other seed edges might be shorter. □

6.3 All Correct Edges are Constructed

Proof (part 2). Every correct edge appears either inside a component of the 1-manifold decomposition or it connects a component with a cut vertex. Since every component contains at least one seed edge, the algorithm will reconstruct every component either starting at its seed edge or coming from a neighboring component over a cut vertex. The components are reconstructed as a whole by lemma 1 including adjacent cut vertices so it is guaranteed that no correct edge is left out. □

6.4 Correctness Theorem

Theorem 1 (Correctness). *Let S be a valid sample of a reconstructible collection Σ. The algorithm in section 4.1 generates exactly all the correct edges possible in S and hence it is a correct reconstruction algorithm by definition 6.1.*

Proof. See lemma 1 and section 6.2 and 6.3. □

7 Data Structures to Improve Efficiency

7.1 Dynamic Closest Pair Maintenance

Points in a *fair-split tree* are not separated by their median like in a kd-tree for example but they are divided fair such that a fixed proportion is in the one subtree and the rest in the other one. Bespamyatnikh [3] uses this balanced tree to allow a certain laziness performing updates and thereby achieves a data structure which dynamically maintains the closest pair of points. The initial tree of size $\mathcal{O}(n)$ can be build in $\mathcal{O}(n \log n)$ time and each operation (insert point, delete point, query closest pair) takes only $\mathcal{O}(\log n)$ time. This approach works for any fixed dimension.

7.2 Probe Inflation with Partition Trees

A partition tree is a data structure which partitions a point set into a constant number r of disjoint subsets and does the same recursively with the subsets. The big advantage of the structure is that the subsets have nearly equal size and they can be enclosed by triangles such that a query line does not intersect more than $\mathcal{O}(r^{1-1/d})$ triangles. Therefore a range query counting the number of points inside a constant sized polygon can be executed in $\mathcal{O}(n^{1-1/d})$ time by recursion over the intersected triangles. A basic explanation of partition trees and comparison with related data structures is given in the survey by Matoušek [14]. The same author also provides a detailed analysis in [15].

Partition trees can be used for probe inflation in the following way. For each of the r triangles in a partition a single point (fixed or random) is chosen and tested whether it lies inside the probe or not. If it does, the probe will become shrunken such that the point now lies on the boundary. This guarantees that no triangle is completely inside the probe, so every one must be completely outside or it intersects the probe. Now the usual algorithm continues.

Non-convex probes are triangulated in advance and the triangles are used for the query.

7.3 Practical Efficiency with kd-Trees

In practice both important queries, nearest unmarked neighbor and point minimizing the distance function, can be executed very fast using only a single simple kd-tree implementation. At least for $k = 2$ dimensions this performs very well for moderate point sets due to very small runtime constants compared to the other advanced data structures proposed.

8 Analysis

8.1 Runtime of the Presented Algorithm

Let n denote the size of the valid sample S. The algorithm consists of the following two main routines which are called very often: Find a seed edge and find a proper consecutive edge for a given edge.

The runtime is composed of these two main steps resulting in a total of $\mathcal{O}(T_p + i \cdot T_i + e \cdot T_e)$ steps for i seed edges and e edges in total with T_p preprocessing time and T_i, T_e denoting the time to find a seed edge or looking for a consecutive edge respectively. Note that for all edges, including seed edges, a consecutive one is searched. Evaluating the function F, setting variables and adding edges to the reconstruction are accounted as constant time operations.

If the number of vertices in each component in the 1-manifold decomposition is bounded by a constant c, at least $\frac{n}{c}$ components exist and therefore $i \in \Theta(n)$ seed edges must be found in the worst case as in figure 13. This already happens for a set of disjoint triangles.

The algorithm reconstructs the collection component-wise with respect to the 1-manifold decomposition. Such a connected component is

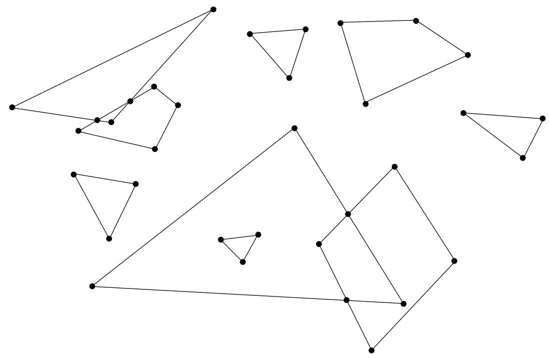


Figure 13: Worst case: Lots of curves with a limited number of vertices

a manifold by definition 5.2 and therefore its reconstruction is a line strip. A closed line strip with n vertices has n edges, an open one has only $n - 1$ edges. At most two additional edges connect the component with *cut vertices*. Let s_i denote the number of vertices per connected component. Since the components are disjoint, $\sum_j s_j \leq n$ holds and one gets the following for the number of edges e.

$$e \leq \sum_j (s_j + 2) < 2n + \sum_j s_j = 3n$$

Every reconstruction has at least $\left\lfloor \frac{n}{2} \right\rfloor$ edges. This is obvious because the algorithm continues to connect two unused points until there is at most one point left. Hence the total number of generated edges is $\Theta(n)$. This also remains true for higher dimensions.

If all is implemented in a brute-force manner then $T_i \in \mathcal{O}(n^2)$ and $T_e \in \mathcal{O}(n)$ and thus the total runtime will become $\mathcal{O}(n^3)$ which can easily be reduced to $\mathcal{O}(n^2 \log n)$ by intelligent handling of the $\mathcal{O}(n^2)$ possible edges.

Using the data structures proposed in chapter 7 can dramatically improve the runtime noted above. The efficient maintenance of the the closest pair requires $\mathcal{O}(\log n)$ time per operation. This makes $T_i \in \mathcal{O}(\log n), T_e \in \mathcal{O}(n + \log n)$ and reduces the total runtime to $\mathcal{O}(n^2)$. An even further reduction of the runtime is achieved by using partition trees for efficient probe inflation. It reduces the query time for a consecutive edge from $\mathcal{O}(n)$ to $T_e \in \mathcal{O}(n^{1-1/d})$, yielding the final runtime of $\mathcal{O}(T_p + iT_i + eT_e) = \mathcal{O}(2n \log n + n^{2-1/d}) =$

$\mathcal{O}\big(n^{2-1/d}\big)$.

Theorem 2. *The algorithm in section 4.1 provides a correct reconstruction by definition 6.1 in* $\mathcal{O}\big(n^{2-1/d}\big)$ *steps for a sample with* n *sample points which is valid by definition 5.4.*

8.2 Space Requirements

Using the described data structures one has to maintain the partition tree and the fair-split tree for the nearest neighbors. Both have linear size in the number of points n. The recognition whether an edge is already in Γ or not can be done by storing the edges in a balanced search tree. This provides a query time of $\mathcal{O}(\log n)$ per found consecutive edge but searching this edge using the partition tree already takes $\mathcal{O}\big(n^{1-1/d}\big)$ which dominates.

This reduces the size of the used storage to $\Theta(n+|\Gamma|)$ where n is the number of points and Γ the size of the output. Since Γ has linear complexity in n as explained in a prior section, the algorithm uses linear space which is optimal.

Theorem 3. *The algorithm in section 4.1 provides a correct reconstruction by definition 6.1 in* $\mathcal{O}\big(n^{2-1/d}\big)$ *steps using* $\Theta(n)$ *space for a sample with* n *sample points which is valid by definition 5.4.*

9 Experimental Results

9.1 Implementations

The presented algorithm was implemented in several ways to test its potential in practice. One implementation was done in Java as a plug-in for the interactive geometry software Cinderella [13] shown in figure 14.

Some brute-force implementations were done using C++ but they gave the expected bad results since the runtime is cubic in the number of points in the worst case.

A very simple implementation using a single kd-tree for the closest pair and the consecutive edge searching was done. In tests with randomly sampled and equally distributed sampled collections it showed a good performance and for small points sets of up to 10000 points the speed-up compared to the brute-force implementation was already a factor of about 100.

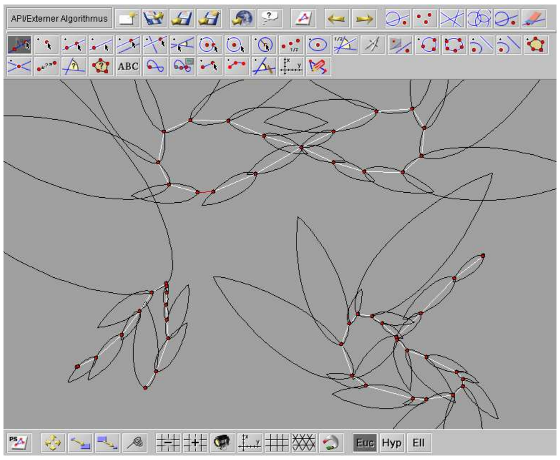


Figure 14: Screenshot from Cinderella with the interactive curve reconstruction plug-in.

The development of good heuristics which take the typical special structure of a sample in the local neighborhood of a point into account is a matter of further research.

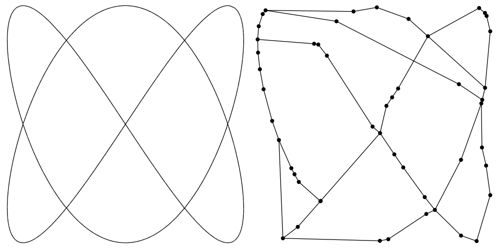
9.2 Heuristic Results

A second series of experiments were done to show the practicability of the algorithm as a heuristic approach. It is not always possible to guarantee the validity of a sample and it is impossible to check without the original curve. In these cases the algorithm should nevertheless give some sensible output.

The Lissajou figure in figure 15(a), given by the parametric form $L(t) \mapsto (\sin 4\pi t, \cos 6\pi t)$, was sampled such that n values were taken equally distributed from $[0; 1)$ and the corresponding points were put into the sample. This was done for several values of n.

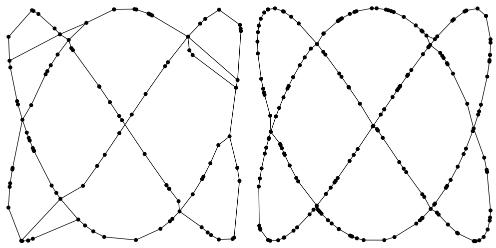
The results depend of course on the random input but they show a general behavior which can be reproduced for other inputs of the same size. One possible drawback of the algorithm is that a single failure—a single wrong edge—can lead to a chain reaction for the following edges because they all base on a wrong edge. The experiments show that this disastrous effect is not likely to occur if the sample has at least a certain density.

Figure 15(b) shows the reconstruction for 50 points which is quite bad and definitively under-



(a) This figure was sampled and reconstructed.

(b) n=50: The sample is overall too sparse.



(c) n=100: Some wrong edges appear resulting from sharp bends.

(d) n=200: Almost perfect result except for minor dents and one wrong edge.

sampled. In figure 15(c) 100 points were taken. Some bends are too sharp to be captured and some problems occur caused by wrong connections around intersections. The result for 200 points in figure 15(d) is already very satisfying and one can clearly see that although a wrong point was taken near the intersection on the left side, the curve continues as expected and the topology is correct. In the upper right corner a wrong edge was reconstructed but only a single one which does not entail other wrong edges. These small glitches might be cleaned by some post-processing.

Obviously one can always create a point pattern around an intersection which will result in a wrong reconstruction independent of the density. Nevertheless there are no additional wrong edges besides close to intersections and no gaps, so the reconstructed topology will be correct if the sample is dense enough. This suggests that the algorithm might also be used successfully in a heuristic approach.

10 Discussion and Conclusion

The presented algorithm follows very simple ideas which are developed and justified throughout the paper. It reconstructs every "known" feature of curves in any dimension without special handling and with few sample points.

Similar to ε-samples the idea of not crossing the medial axis is contained in the placing restriction of the points but it is handled more flexible because in some places crossing the medial axis is explicitly necessary, e.g. at corners without the corner point in the sample. The sample is not limited by geometric properties of the original curve which allows to emphasize details in certain regions and neglect others by adjusting the sampling density.

On the downside, the runtime is $\mathcal{O}(n^{1.5})$ in two dimensions which is worse than the $\mathcal{O}(n \log n)$ of many other algorithms but since the algorithm introduced here relies on a different sampling condition which also can reconstruct collections from sparse samples it is like comparing apples and oranges.

References

[1] E. Althaus and K. Mehlhorn. Polynomial time TSP-based curve reconstruction. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 686–695, Jan. 2000.

[2] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60:125–135, 1998.

[3] S. N. Bespamyatnikh. An optimal algorithm for closest pair maintenance. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 152–161, 1995.

[4] D. Cohen-Steiner and F. Da. A greedy delaunay based surface reconstruction algorithm. Rapport de recherche 4564, INRIA, 2002.

[5] T. K. Dey and P. Kumar. A simple provable algorithm for curve reconstruction. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 893–894, Jan. 1999.

[6] T. K. Dey, K. Mehlhorn, and E. A. Ramos. Curve reconstruction: Connecting dots with good reason. *Comput. Geom. Theory Appl.*, 15:229–244, 2000.

[7] T. K. Dey and R. Wenger. Reconstructing curves with sharp corners. *Computational Geometry Theory Applications*, 19:89–99, 2001.

[8] S. Funke. *Combinatorial Curve Reconstruction and the Efficient Exact Implementation of Geometric Algorithms*. PhD thesis, Universität des Saarlandes, Postfach 151150, D-66041 Saarbrücken, Germany, 2001.

[9] S. Funke and E. A. Ramos. Reconstructing a collection of curves with corners and endpoints. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 344–353. Society for Industrial and Applied Mathematics, 2001.

[10] J. Giesen. Curve reconstruction in arbitrary dimension and the traveling salesman problem. In *Proc. 8th Int. Conf. Discrete Geometry for Computer Imagery*, pages 164–176, 1999.

[11] J. Giesen. Curve reconstruction, the TSP, and Menger's theorem on length. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 207–216, 1999.

[12] M. Knobelsdorf. Kurvenrekonstruktion mit Schnittpunkten. Master's thesis, Freie Universität Berlin, Kaiserswerther Str. 16–18, 14195 Berlin, Germany, September 2004.

[13] U. Kortenkamp and J. Richter-Gebert. Cinderella. `http://www.cinderella.de`.

[14] J. Matoušek. Geometric range searching. Tech. Report B-93-09, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 1993.

[15] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993.