Simple Reconstruction of Non-Simple Curves

and

Approximating the Median in Streams with Constant Storage

Dissertation zur Erlangung des Doktorgrades

vorgelegt am

Fachbereich Mathematik und Informatik

der Freien Universität Berlin

2008

von

Tobias Lenz

Institut für Informatik
Freie Universität Berlin
Takustraße 9
14195 Berlin
tlenz@mi.fu-berlin.de

Betreuer: Prof. Dr. Günter Rote
Institut für Informatik
Freie Universität Berlin
Takustraße 9
14195 Berlin
rote@mi.fu-berlin.de

Gutachter: Prof. Dr. Günter Rote
Institut für Informatik
Freie Universität Berlin
Takustraße 9
14195 Berlin
rote@mi.fu-berlin.de

Prof. Dr. Stefan Funke
Institut für Mathematik und Informatik
Universität Greifswald
Jahnstraße 15a
17487 Greifswald
stefan.funke@uni-greifswald.de

Datum der Disputation: 28.10.2008

CONTENTS

LIST OF ALGORITHMS

DANKSAGUNG

Ich bedanke mich bei allen, die bei der Entstehung dieser Dissertation geholfen haben. Allen voran bedanke ich mich bei Günter Rote für die Betreuung und die konstruktiven Gespräche. Ich danke Stefan Funke für seine Tätigkeit als Gutachter.

Mein Dank gilt der Arbeitsgruppe *Theoretische Informatik* der Freien Universtität Berlin für ein unübertreffliches Arbeitsklima. Insbesondere bedanke ich mich bei den Kollegen Britta Broser und Klaus Kriegel für die Hilfe beim zweiten Teil der Arbeit.

Des Weiteren bedanke ich mich bei meiner Familie und meinen Freunden für die Unterstützung während meiner Zeit als wissenschaftlicher Mitarbeiter.

Part I

SIMPLE RECONSTRUCTION OF NON-SIMPLE CURVES

ABSTRACT

This work generalizes the ideas in the Nearest-Neighbor-Crust algorithm by Dey and Kumar. It allows to reconstruct smooth, closed curves from ε-samples with $\varepsilon \leq 0.48$. This is a big improvement compared to the original bound. Further generalization leads to a new algorithm which reconstructs closed curves with self-intersections. The algorithm is very simple and short and works well in practice. A special ε-sampling condition is given which guarantees correct results. The described method works for curves in any dimension d in $O\big(n^{2-1/d}\big)$ time.

ZUSAMMENFASSUNG

In diesem Kapitel wird die Idee des Nearest-Neighbor-Crust-Algorithmus von Dey und Kumar verallgemeinert. Das Ergebnis ist ein Algorithmus, der die Rekonstruktion von glatten, geschlossenen Kurven aus einem ε-Sample mit $\varepsilon \leq 0,48$ erlaubt. Dies ist eine gravierende Verbesserung gegenüber der Schranke für den unveränderten Algorithmus. Weitere Verallgemeinerung führt zu einem neuen Algorithmus, der geschlossene Kurven mit Selbstschnitten rekonstruieren kann. Der vorgestellte Algorithmus ist sehr einfach und kurz und in der Praxis einsetzbar. Es wird eine spezielle Bedingung für das ε-Sample angegeben, unter der der Algorithmus beweisbar korrekte Ergebnisse liefert. Das Verfahren funktioniert für Kurven in beliebiger Dimension d in $O\big(n^{2-1/d}\big)$ Zeit.

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" [DMR00]. The "dots" are a finite set of points, the so called *sample*, taken from a curve. The task is to connect them retaining the adjacency of the original curve without knowing this curve. The outcome should be a polygonal approximation of the original curve which the samples were taken from. The approximation quality is not measured with some error metric but only with respect to the correct adjacencies, so either the reconstruction is correct or it is not. Hence it is in fact "connecting the dots" with the goal to find the correct order.

sample

This is exactly the task the computer should solve for us. Of course the results depend on the positions of the given points and the used algorithm to connect them. So there is a natural demand for a classification of curves by certain features which make them easy to reconstruct or hard or maybe even impossible—why is it for example much easier to reconstruct a duck than a fish? (Figure 1.1)

1.2 Applications

A major application for reconstruction in practice is the creation of surfaces from point clouds generated by 3d scanners. This is the *surface reconstruction* problem. Curve reconstruction provides foundations and new ideas for surface reconstruction but also has its own applications.

surface reconstruction

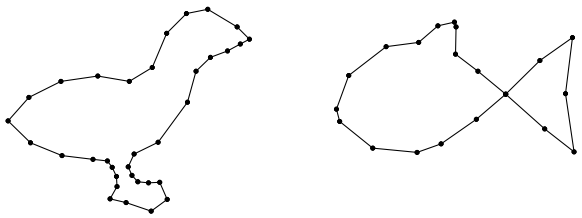


Fig. 1.1: A reconstruction of a duck and a fish. Although they look simple, both figures have features which are difficult to reconstruct: sharp features (turns more than $\pi/2$) and a self-intersection

Althaus [Alt01] describes several image processing applications. Looking for a shape, an image is scanned for characteristic points. Due to image quality, the shape's complete boundary usually cannot be detected, so some regions of uncertainty remain. A curve reconstruction algorithm can connect the pieces. This technique is most important for medical applications like computer tomography and magnetic resonance imaging. A 3d shape of a body part is represented as several layers of 2d images. In each of these slices the boundary of e.g. a bone is reconstructed as a polygonal curve. Afterward the polygons from each slice are connected to the final 3d shape.

Another application is the digitization of road maps [GS01]. It requires topological correctness, otherwise a street on the polygonal map might lead through a lake or over a cliff. The traditional method to fix the topology was to place points right inside of each area along the boundary and label them respectively. Then the Voronoi edges between points with different labels are considered as reconstruction. The placement of points and labeling was done manually or semi-automatically. Using curve reconstruction, the whole process can now run fully automated because many reconstruction algorithms come with guarantees on the topology.

implicit curve

A mathematical application which does not involve scanned images of any kind is to approximately plot *implicit curves*. These are defined as the zero set of a function $f(x, y)$. Only for very simple functions f, a transformation into a parametric curve is possible. An implicit curve can contain several connected components. It need not be smooth and can contain saddle points. Algorithms like marching cubes [LC87] try to do an exhaustive case distinction on small regions of the domain. A simple alternative is to approximate several roots of f by Newton's method and apply a curve reconstruction algorithm.

1.3 Sampling Conditions

sampling condition

The quality of curve reconstruction depends heavily on the quality and quantity of the input points. This is expressed in a so called *sampling condition* under which a certain algorithm guarantees a correct reconstruction.

All sampling conditions limit the distance between two samples which are consecutive on the curve. How this limitation is achieved varies widely. Some known sampling conditions are: uniform sampling, ε-sampling, a condition with respect to the correct reconstruction instead of the original curve [Fun01, FR01], and a condition based on visibility regions [Fre02]. These conditions 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 rule is made very explicit in the ε-sampling condition which is used in this thesis as well as in many other publications. In general a special sampling condition suits best for a special algorithm, providing a sufficient condition for a correct

reconstruction.

It is apparently easier to reconstruct from a sample which is dense everywhere, than from a sparse one. Therefore it is not only important to find a proper sampling condition, but also to find reasonable constants for it. In the case of an *ε-sample*, the sample becomes sparser for larger values of ε, so we look for a preferably large ε. *ε-sample*

1.4 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. If an algorithm only handles simple curves (without intersections), the reconstruction is a planar graph with n edges. This allows to pre-select a linear number of the $\Theta(n^2)$ possible edges and then throw out edges step by step until the remaining edges form the final reconstruction. It has been shown in several papers [ABE98, DK99, DMR00, DW01, FR01, DW02] that the edges of the Delaunay triangulation are appropriate for this filtering. Further advantages are that they can be computed efficiently and the concept naturally extends to higher dimensions.

The success of the Delaunay edges is coupled to the requirement for an *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 to curve reconstruction by Amenta, Bern and Eppstein [ABE98] to distinguish between smaller features needing a high sample density and larger features which only need a low sample density to be captured well. *ε-sample* *local feature size* *medial axis*

1.5 Known Reconstruction Algorithms

Some of the milestone results for curve reconstruction are listed here. Due to the large number of heuristics and small modifications to known algorithms, this list cannot be complete.

In 1983, Edelsbrunner, Kirkpatrick, and Seidel introduced *α-shapes* [EKS83]. They create a ball around each sample with weighted radius and consider the intersection of the Delaunay triangulation of the samples with the union of these balls as reconstruction. For specific radii depending on a parameter α, this intersection is shown to be a correct reconstruction. The concept naturally extends to higher dimensions and is therefore also used for surfaces. *α-shape*

In 1998, Amenta, Bern, and Eppstein presented the Crust algorithm [ABE98]. The main idea is that reconstruction edges should not intersect the *medial axis* of the original curve. The medial axis is not known but the authors *medial axis*

showed that a subset of the Voronoi edges of the sample points approximates the medial axis if the sample is dense enough. They proof that their algorithm works for ε-samples with $\varepsilon \leq 0.252$. This result was improved to $\varepsilon < 1/3$ by Dey and Kumar with the Nearest-Neighbor-Crust algorithm [DK99]. This algorithm is described in detail in Section 3.3.1.

The first reconstruction of open curves was provided by Dey, Mehlhorn, and Ramos in 2000 [DMR00]. For their Conservative Crust algorithm they introduced a parameter ρ which controls how easy an edge of the Delaunay triangulation is filtered out.

Dey and Wenger designed a heuristic called Gathan in 2001 [DW01] which was able to reconstruct closed curves with sharp corners. Later they modified the sampling condition to obtain provable results [DW02].

An algorithm which combines the strengths of the Conservative Crust and Gathan was introduced by Funke [Fun01], respectively Funke and Ramos [FR01]. They question the requirement for the dense ε-sample if a close approximation of the original curve is not needed. They define a new sampling condition which allows sparse samples even for feature-rich curves. Additional conditions about the position of the samples relative to each other guarantee the reconstructability. Correct results are proven for collections of open curves with sharp corners.

1.6 Our Contribution

An ε-sample does not exist for curves with sharp corners or intersections. As a minor modification to the ε-sampling condition, we exclude small parts of the original curve. We present a simple algorithm with several parameters and provide a choice of these parameters which reconstructs smooth, closed curves for $\varepsilon \leq 0.48$. Another set of parameters of the same algorithm is provided which allows to reconstruct smooth, closed curves with self-intersections or collections of several curves which intersect each other for $\varepsilon \leq 0.138$. Some parts of this thesis have already been published [Len05a, Len05b, Len06b].

1.7 Related Results

traveling salesperson problem

The problem of reconstructing closed curves might also be formulated as an instance of the well known *traveling salesperson problem* as done by Giesen [Gie99a, Gie99b]. 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 [AM00, Alt01] showed that this tour can be found in polynomial time although the general TSP problem is NP-hard.

Other approaches use additional information, e.g. normals for the sample points like the so called tensor voting [MLT00, LZJ+05] which allows reconstruction from very noisy data.

The surface reconstruction by Cohen-Steiner and Da [CSD02] uses a greedy technique growing the reconstruction adding the most fitting triangles one at a time which is roughly similar to the idea proposed in this thesis for curves. They start with a triangle and add more triangles to the surface by evaluating some topological properties of the resulting surface, e.g. a triangle that closes a hole is a much better choice than to create intersecting triangles. This evaluation is done for curves in this thesis by a simple "probe" function which bases on the turning angle.

2. PREREQUISITES

In curve reconstruction, the task is to find a polygonal approximation of an original curve based on a preferably small sample of that curve. The curve and sample are defined as follows.

Definition 2.1. A mapping $\sigma : A \to B$ is called *locally injective* if for all $a \in A$ a non-empty neighborhood $N(a) \subset A$ of a exists such that $\sigma : N(a) \to B$ is injective. *locally injective*

Definition 2.2. Let σ be a continuous and locally injective mapping $\sigma : A \to \mathbb{R}^d$. We denoted the image of σ by Σ. We call Σ a (d-dimensional) *closed curve*, if A is the one-dimensional sphere. For $A = [0, 1]$ and $\sigma(0) \neq \sigma(1)$ we call Σ an *open curve*, and $\sigma(0)$ and $\sigma(1)$ are called *endpoints*. *closed curve* *open curve* *endpoint*

Sometimes the curve is defined as the image of an injective mapping but since we want to reconstruct self-intersecting curves, we cannot forbid identical images for different parameters.

Definition 2.3. A point p on a curve is called *self-intersection* if $a, b \in A$ exist such that $a \neq b$ and $\sigma(a) = \sigma(b) = p$. A curve is called *simple* if it contains no self-intersections. *self-intersection* *simple curve*

A point $\sigma(c), c \in A$, on a curve is called *corner* if $\sigma(c)$ is not an endpoint and σ has no (unique) tangent at c. A curve is called *smooth* if it has no corners. *corner* *smooth curve*

Figure 2.1 gives an intuition about some of the features i.e. smooth, open and closed curves, endpoints, corners, and intersections.

Definition 2.4. A *sample* S is a finite subset of a curve Σ. The elements of the sample are points in $\mathbb{R}^d$ and are called *sample points*. *sample* *sample point*

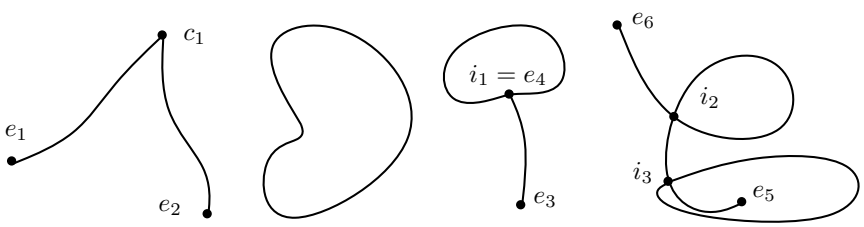


Fig. 2.1: Curve examples with endpoints e_j, corners c_j, and intersections i_j.

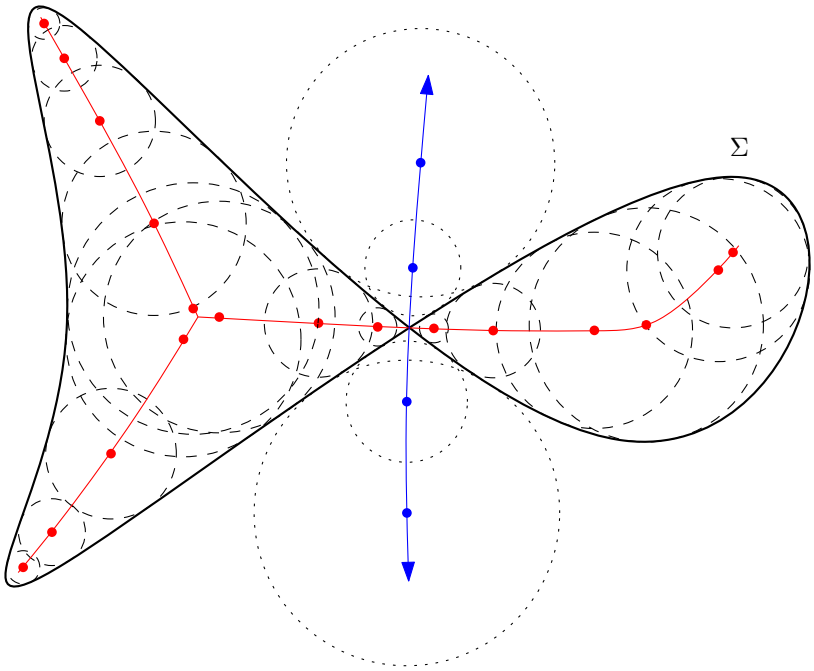


Fig. 2.2: A curve Σ with some medial balls, touching Σ at least twice. The centers of the dashed circles lie on the inner part of the medial axis (red). The outer part of the medial axis stretches to infinity and is therefore only partly visible (blue).

The sample points should fulfill some conditions to guarantee a reconstruction with a certain quality. The very common ε-sampling condition states that the sample should be dense if the curvature is high or parts of the curve are close together while it might be sparse in straight regions of the curve. Formally it is defined using the medial axis, see Figure 2.2 for an example.

medial axis

Definition 2.5. The *medial axis* of a curve is the closure of the set M containing the center points of all empty balls which touch the curve in more than one point. Such a ball is called *medial ball*. The *local feature size* of a point $p \in \Sigma$ is defined as $\mathrm{lfs}(p) = \min_{m \in M} \|p - m\|$.

medial ball
local feature size

ε-sample

Definition 2.6. A sample S is an *ε-sample* of the curve Σ if $\forall p \in \Sigma : \exists s \in S : \|p - s\| \leq \varepsilon \mathrm{lfs}(p)$.

All known reconstruction algorithms which use ε-samples give guarantees only for some ε less than $1/2$. Amenta et al. [ABE98] showed that reconstruction from a 1-sample can be ambiguous. They used the hand drawn example shown in Figure 2.3 as a proof. No results are known for the long standing gap between $1/2$ and 1.

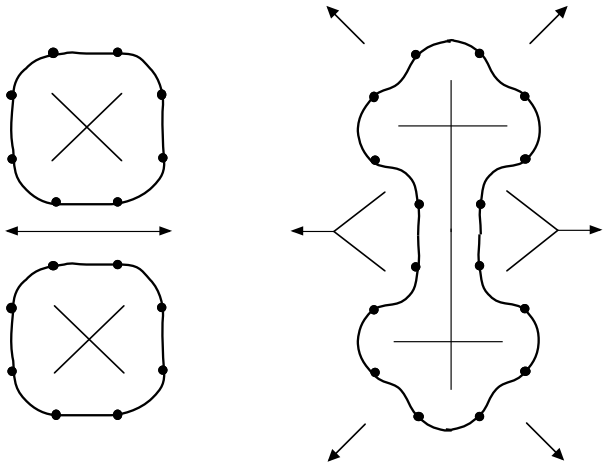


Fig. 2.3: This picture was directly taken from Amenta et al. [ABE98]. It shows their example of a 1-sample which does not have a unique reconstruction. On the left and on the right you see two bold curves with their medial axis. The marked 1-samples are identical.

From the definition of an ε-sample, it is easy to make the following observation.

Observation 2.1. *Every ε-sample is also a ξ-sample for all $\xi > \varepsilon$.*

This simplifies many of the upcoming proofs because they only need to be done for a fixed, sufficiently large ξ and automatically hold for every smaller ε.

The typical way of proving things about ε-samples is by contradiction. If it is possible to bound the distance to the closest sample from below and the distance to the medial axis from above, this implies a bound for ε from below. Choosing an ε smaller than this bound creates the desired contradiction. Therefore it is important to find nearby medial axis points, for example using one of the following lemmas.

Lemma 2.2. *Consider the intersection I of the curve Σ with a closed ball B. If I has at least two connected components, B contains a medial axis point. If one of the components is only a single point x on the boundary of B, then the local feature size of x is at most the radius of B.*

Proof. Continuously shrink B around its center until it contains only one component C in its interior and touches at least one on the boundary. Now fix one of the touching points $t \notin C$ on the boundary and shrink the ball further around t until C is also on the boundary. This is possible because C is continuous. The shrunken ball is an empty ball with at least two

components on its boundary as required in Definition 2.5. Hence its center is a medial axis point and because the shrunken ball stays in the interior of B all the time, B contains a point of the medial axis.

If one of the components is only a single point x on the boundary of B, we can ignore the initial shrinking step and start with $t = x$. The distance from t to the center of B is the radius and it might only become smaller because B might get shrunken further while still touching x. □

Lemma 2.3. *Consider the intersection I of the curve Σ with a closed ball B. On every diameter d of B which touches or stabs at least two of the connected components in I, there is a medial axis point.*

Proof. Figure 2.4 illustrates the idea of this proof.

We call the first two distinct connected components stabbed by d Σ_1 and Σ_2. We move a point x continuously along a subset of d from an arbitrary intersection point of d and Σ_1 to an arbitrary intersection point of d and Σ_2. We keep track of the distances f_1, f_2 between x and its closest point in Σ_1 and Σ_2 respectively. The distance between x and the closest point on any other connected component in B is denoted by f_3. If Σ_1 and Σ_2 are the only components in B, f_3 is set to infinity.

Since x starts on Σ_1, we have $f_1 = 0, f_2 > 0, f_3 > 0$ in the beginning. In the end we have $f_1 > 0, f_2 = 0, f_3 > 0$ because x ends on Σ_2. While moving x along d, f_1, f_2, and f_3 change continuously. Therefore $f_1 = f_2 < f_3$ or $f_1 = f_3 < f_2$ or $f_2 = f_3 < f_1$ follows for some x on d from the intermediate value theorem. This x is a medial axis point by Definition 2.5. □

Lemma 2.4. *Every point on the curve is adjacent to its nearest sample point in an ε-sample with $\varepsilon < 2$.*

Proof. Assume p is the sample point closest to x and they are not adjacent on the curve. Further assume that p and x are the closest pair with this property. Consider a ball B centered at $(p + x)/2$ with radius $\|p - x\|/2$. The intersection of the curve part around x with B must be x, otherwise B contains points closer to p with the demanded property. Lemma 2.2 states that a medial axis point is not farther away from x than the radius $\|p-x\|/2$.

Let q be a closest sample point adjacent to x. By Definition 2.6 we have $\|x - q\| \leq \varepsilon\|p - x\|/2 < \|p - x\|$. This contradicts the assumption. □

turning angle

Definition 2.7. The *turning angle* $\sphericalangle(p, q, r)$ of three points p, q, r is defined as the absolute value of the smallest rotation angle needed to transform the oriented line $\overline{pq}$ into the oriented line $\overline{qr}$. See Figure 2.5 for an illustration.

inner angle

The turning angle is always between 0 and π. Note that this angle differs from the *inner angle* $\angle(p, q, r)$ in q of the triangle defined by p, q, r which is usually associated with three points. In fact $\sphericalangle(p, q, r) = \pi - \angle(p, q, r)$,

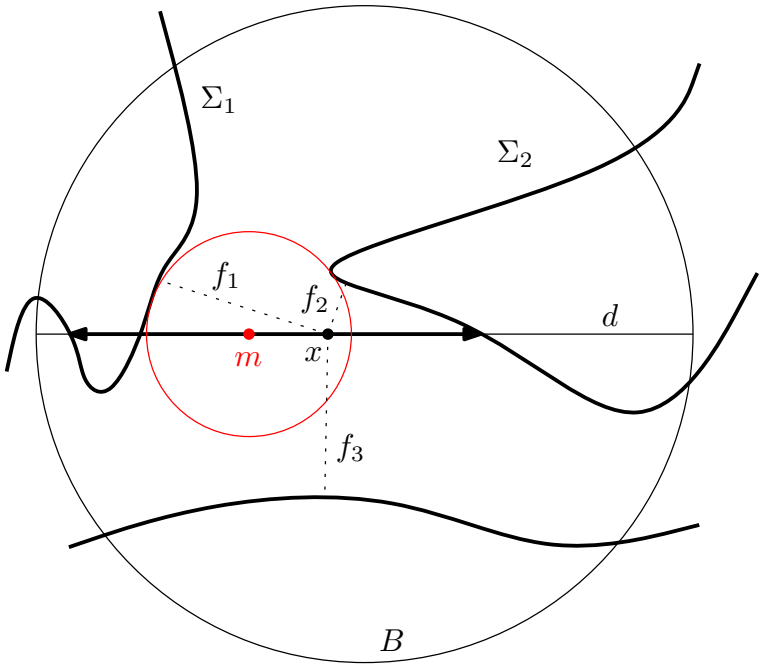


Fig. 2.4: The connected components Σ_1, Σ_2 are stabbed by the diameter d. The point x moves along the bold subset of d and the distances f_1, f_2, f_3 are monitored. Point m is a medial axis point, because the f_i change continuously and for $x = m$ we have $f_3 > f_1 = f_2$.

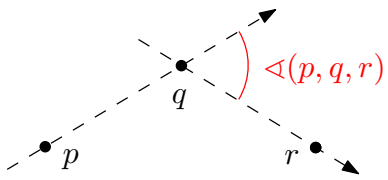


Fig. 2.5: Three points p, q, r and their turning angle $\sphericalangle(p, q, r)$ indicated in red.

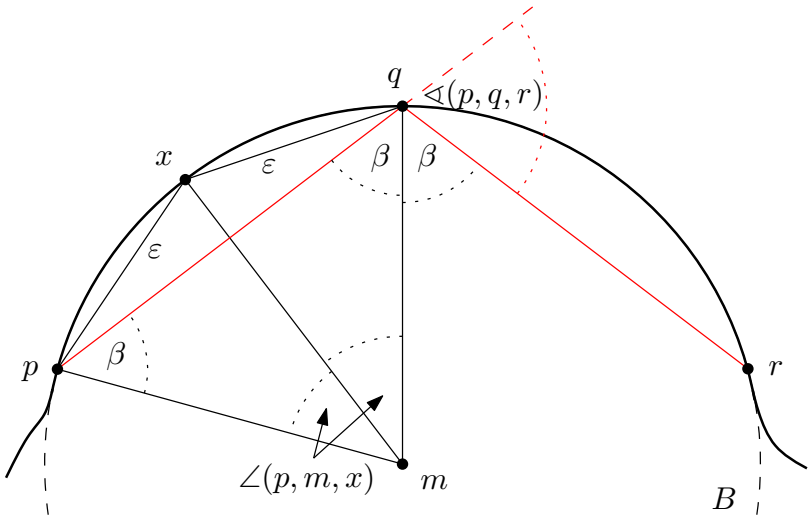


Fig. 2.6: An illustration of the proof for Lemma 2.5. The maximum turning angle is obtained by putting three consecutive samples on a medial ball.

but the turning angle is an important and intuitive quantity throughout this thesis, so it has its own symbol.

The next lemma is used in many papers about curve reconstruction. It was first proven by Amenta et al. [ABE98, Lemma 10] using a quite different terminology based on Voronoi diagrams. We do not repeat their proof in full length here but we show the last part of it in our own terminology after the lemma.

Lemma 2.5. *The turning angle of three adjacent samples in an ε-sample with $\varepsilon < 1$ is at most $4 \arcsin(\varepsilon/2)$. Applying elementary transformations to this yields the following result: To achieve a turning angle for any three adjacent samples from an ε-sample of at most $\tau \leq \pi/2$, one can choose $\varepsilon \leq 2 \sin(\tau/4)$.*

Proof (part). The proof of Amenta et al. [ABE98, Lemma 10] ends up with a curve along the boundary of an empty circle B.

Consider three adjacent samples p, q, r from an ε-sample of B. The center of B is a medial axis point m because B is empty. The local feature size of any point on B is $\|q - m\|$, the radius of B. This situation is depicted in Figure 2.6.

Consider the point $x \in \Sigma$ which has the same distance to p and q. For an ε-sample $\|x - p\| \leq \varepsilon \mathrm{lfs}(x)$ and $\|x - q\| \leq \varepsilon \mathrm{lfs}(x)$ holds. By the definition of sine we get

$$\frac{\|x - p\|}{\|q - m\|} = 2 \sin \frac{\angle(p, m, x)}{2} \leq \varepsilon$$

which can be turned into

$$\angle(p, m, x) \leq 2 \arcsin \frac{\varepsilon}{2}.$$

Due to symmetry we have $\angle(p, m, x) = \angle(p, m, q)/2$ and because we are in a right triangle, $\beta = \pi/2 - \angle(p, m, q)/2$ holds. These equations also hold for the other side with r instead of p. Therefore $\sphericalangle(p, q, r) = \pi - 2\beta$. Putting all this together yields

$$\begin{aligned}
\sphericalangle(p, q, r) &= \pi - 2\beta \\
&= \pi - 2\frac{\pi - \angle(p, m, q)}{2} \\
&= \angle(p, m, q) \\
&= 2\angle(p, m, x) \\
&\leq 4 \arcsin \frac{\varepsilon}{2}.
\end{aligned}$$

Note that the result is independent of the radius of B.

3. THE NEW APPROACH

3.1 "Seed Edges" and "Probe Inflation"

The idea is to inductively grow the reconstruction, one edge after the other. This is done in a greedy fashion. A special shape, called *probe*, is aligned along an existing edge. Then the probe becomes inflated until it hits a sample point. This defines a new edge. Figure 3.1 illustrates several steps of the described process. *probe*

Since every vertex in a reconstruction of a smooth closed curve has degree two, one only has to find a single *seed edge* and grow the reconstruction from that. The overall shortest edge is used as seed edge throughout this thesis. It is a correct edge for ε-samples with $\varepsilon < 1/2$, as shown in the following lemma. Other criteria for seed edges are possible, e.g. manually picked edges, but the shortest edge seems to be most useful. *seed edge*

Lemma 3.1. *In an ε-sample with $\varepsilon < 1/2$, the closest sample point pair forms a correct edge.*

Proof. The proof for two dimensions is illustrated in Figure 3.2. Let p, q be the closest sample point pair. This implies that the union U of the balls around p and q with radius $\|p - q\|$ must not contain samples. Consider the ball B with diameter $\overline{pq}$. B lies completely in U. If the edge (p, q) is not correct, at least one sample between p and q must exist, which cannot lie in B, so B contains two parts of the curve, one containing p and one containing q respectively. Thereby a medial axis point m must lie on $\overline{pq}$ by Lemma 2.3. Without loss of generality $\|p - m\| \leq \|q - m\|$ holds and therefore also $\|p - m\| \leq \|p - q\|/2$.

Choose a point x with $\|x - p\| = \|p - q\|/2$. This point has a distance $\|x - m\| \leq \|x - p\| + \|p - m\| \leq \|p - q\|$ to the closest medial axis point m by triangle inequality. For an ε-sample we have $\|x - p\| \leq \varepsilon\|x - m\|$, hence $\|p - q\|/2 \leq \varepsilon\|p - q\|$. This does not hold for $\varepsilon < 1/2$. □

Note that this proof is independent of the dimension. The result holds for every $\varepsilon < 1/2$ but the largest value which is used for ε throughout this thesis is 0.48, so this restriction on ε is negligible.

The name *probe* refers to the shape one obtains from drawing a proper function θ in polar coordinates for the full circle with θ as distance from the origin. The probe is not defined for negative values because the *turning* *probe* *turning angle*

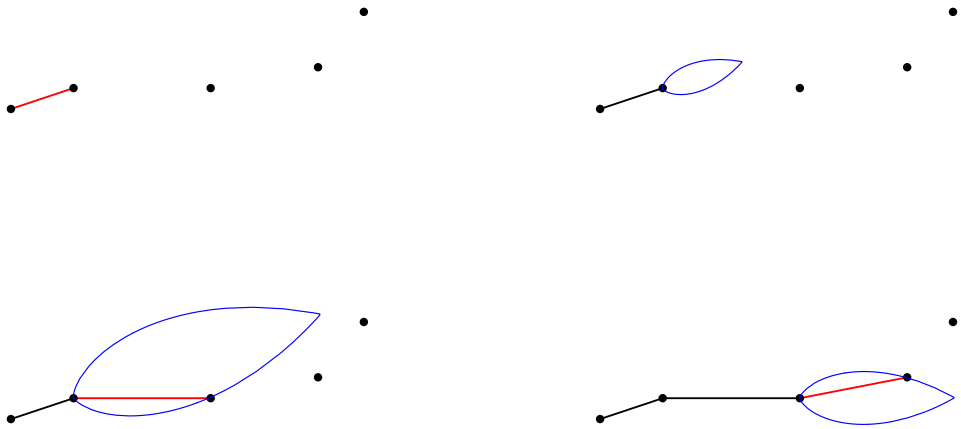


Fig. 3.1: The basic steps of probe inflation are illustrated: Pick a seed edge (top left), align the probe along the edge (top right), inflate the probe until a sample point is hit (bottom left), continue the process (bottom right).

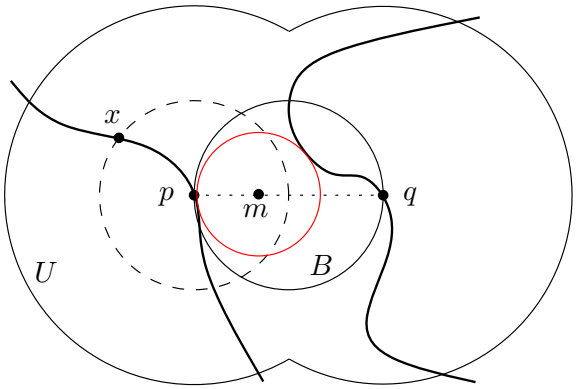


Fig. 3.2: The closest sample pair forms a correct edge.

angle is non-negative. This makes the probe symmetric.

In general more or less every function in one parameter could define a probe. Continuity, symmetry, or monotonicity are technically not necessary. Nevertheless the following definition defines a probe on some kind of monotonicity and implicit symmetry. It is a sensible definition with respect to curve reconstruction.

Definition 3.1. A map $\theta : [0, \alpha] \to \mathbb{R}_{\geq 0}$ is called *α-probe* for $0 < \alpha \leq \pi$ if it is monotone decreasing. *probe*

An α-probe θ has *negative extent* if $\alpha = \pi$ and $\theta(\pi) > 0$.

Figure 3.3 shows shapes which are probes by Definition 3.1. To allow the usage of simple, efficient data structures, the probe should be convex and for easy computations also polygonal.

Inflating a probe is mathematically equivalent to minimizing the following distance function.

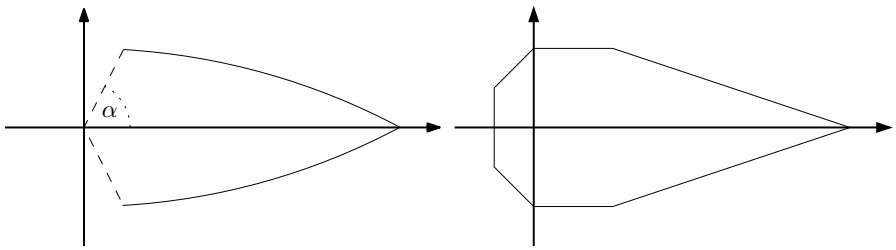
Definition 3.2. Let θ be an α-probe by Definition 3.1. Its corresponding *probe distance function* for three points p, q, r is defined as *probe distance function*

$$D_{pq}(r) = \begin{cases} \dfrac{\|q - r\|}{\theta(\sphericalangle(p, q, r))} & \text{if } \sphericalangle(p, q, r) \leq \alpha \text{ and } \theta(\sphericalangle(p, q, r)) \neq 0 \\ \infty & \text{otherwise.} \end{cases}$$

This definition is valid for any dimension since only the plane spanned by p, q, r is considered and $D_{pq}(r)$ is computed in that plane. Note that $D_{pq}(r)$ is strictly positive. In general $D_{pq}(r)$ differs from $D_{qp}(r)$.

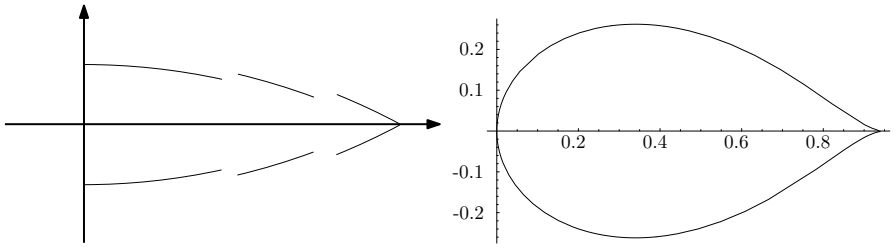
3.2 The New Algorithm

We build the reconstruction edge by edge which is a very local concept. Extending a graph at its "endpoints" with minimum weight edges is a characteristic procedure for greedy algorithms. This idea is in line with famous algorithms like Dijkstra's shortest path algorithm or Prim's algorithm to construct a minimum spanning tree.



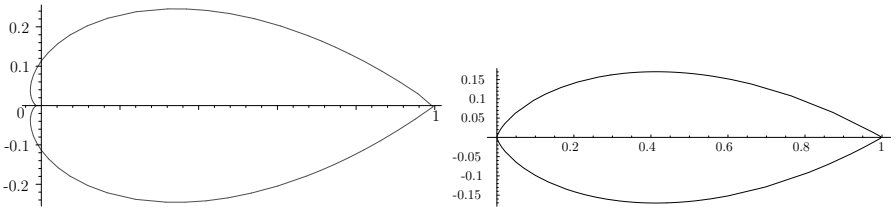
(a) α-probe, convex, without negative extent

(b) polygonal, convex, symmetric, with negative extent



(c) discontinuous $\pi/2$-probe

(d) $\pi/2$-probe with $\theta(\alpha) = \cos\left(\sqrt{|\alpha|} + \pi/2 - \sqrt{\pi/2}\right)$



(e) π-probe with $\theta(\alpha) = 4^{-|\alpha|}$ and negative extent

(f) $\pi/2$-probe with $\theta(\alpha) = \dfrac{\pi/2 - \alpha - \cos\alpha}{\pi/2 - 1}$

Fig. 3.3: Six probes illustrating the variety of possible shapes. All these probes can be computed easily. Probe (a) is assembled from two circular arcs, (b) is a polygon with seven vertices, (c) is a discontinuous combination of circular arcs, (e) is an exponential function of the angle, (d) and (f) are other functions.

Input: Valid ε-sample S from a curve Σ
Output: Polygonal reconstruction Γ of Σ

```
1  Γ ← ∅
2  while |S \ Γ| ≥ 2 do
3  |   find shortest edge/seed edge (p, q) with p, q ∈ S \ Γ
4  |   Γ ← Γ ∪ {(p, q)}
5  |   processEdge (p, q)
6  |   processEdge (q, p)

   Procedure processEdge (p, q) :
7  |   find r ∈ S \ {q} which minimizes D_pq(r)
8  |   if such an r exists and (q, r) ∉ Γ then
9  |   |   Γ ← Γ ∪ {(q, r)}
10 |   |   processEdge (q, r)
```

Algorithm 1: Framework for the reconstruction using probes.

The algorithm starts with a *seed edge*—here the shortest edge available— in line 3. This edge (p, q) becomes part of the reconstructions and the procedure `processEdge` is called for this edge. In the recursive procedure the point r minimizing D with respect to a given edge and orientation (p, q) is found, as defined in Definition 3.2. 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. After the recursion `processEdge` is called for the reversed seed edge to grow the reconstruction in the opposite direction. The reconstructed edges are treated as undirected edges.

seed edge

After a single pass through the while-loop, a single curve is reconstructed. All proofs in this thesis refer to this version of the algorithm, in which a single curve is reconstructed. For more than one curve, the process can be repeated. The while-loop reconstructs curves 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. Therefore the algorithm is able to reconstruct several curves from a single unmarked sample.

The practical issues and possible results of this algorithm are discussed in Section 6.2, page 61. An animated demonstration and an interactive version to experiment with the algorithm can be found in [Len05a] and under `http://www.inf.fu-berlin.de/inst/ag-ti/software/curverec`.

The presented general framework for algorithms using probes can be adapted to specific needs. Finding *seed edges* and finding consecutive edges can 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 distance function D. These rules might even change dynamically.

seed edge

3.3 Compatibility with ε-Samplings: NN-Crust Revisited

As a demonstration that the described algorithm provides sensible results, we will show that the NN-Crust [DK99] algorithm is contained as a special case. The new algorithm is much more flexible which results in better bounds for the required sampling density.

3.3.1 The NN-Crust Algorithm

The *Nearest-Neighbor-Crust* algorithm is an algorithmically simple reconstruction algorithm for curves presented by Dey and Kumar [DK99]. They start with a point set and connect each point p to its nearest neighbor u and its nearest *half-neighbor* h. Consider the line perpendicular to $\overline{pu}$ through p. It splits the plane into two half-planes. The half-neighbor h of p is now the point closest to p which does not lie in the half-plane containing u. This procedure results in a set of edges forming the reconstruction.

half-neighbor

If the underlying point set is a 1/3-sample of a smooth closed curve, the NN-Crust algorithm guarantees a correct reconstruction. Dey and Kumar showed that the edges in the reconstruction are a subset of the edges of a Delaunay triangulation of the point set. This allows simple implementations with $O(n \log n)$ running time in 2d and extensions to higher dimensions.

3.3.2 Generalization of the NN-Crust Algorithm

Changing the way of describing the half-neighbor slightly allows a simple generalization. A similar definition to the one given in Section 3.3.1 is the following: The *half-neighbor* of a point p with nearest neighbor u in a point set S is the point $h \in S \setminus \{p\}$ which minimizes the distance between p and h and fulfills $\sphericalangle(u, p, h) \leq \pi/2$. Now the angle appears as a parameter and instead of $\pi/2$ one can choose an arbitrary angle. These points are no longer called half-neighbors but *α-neighbors* instead, where α specifies the maximally allowed turning angle going from u to p to h. See Figure 3.4 for an illustration. This can be considered as using an α-probe θ with $\theta(x) = 1$ for all x. For the original NN-Crust algorithm we have $\alpha = \pi/2$.

half-neighbor

α-neighbors

The remainder of this section is used to show that for α-neighbors with $0 < \alpha \leq \pi/2$, the resulting edges are still a correct reconstruction. The proof refines and extends ideas from the original paper and leads to a considerably better bound for ε.

Figure 3.5 shows how to bound ε from below using the following arguments.

Lemma 3.2. *Given three adjacent sample points p, q, r in an ε-sample with $\varepsilon < 1$, the curve segment between q and r runs completely inside the cone at q aligned with (p, q) with opening angle $2\alpha = 8 \arcsin(\varepsilon/2)$.*

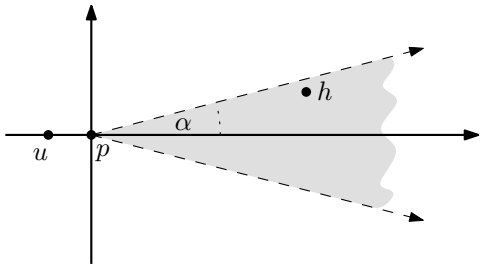


Fig. 3.4: The α-neighbor h of p with nearest neighbor u must lie inside the marked region.

Proof. Assume otherwise and we have a valid ε-sample. Add a sample point on the curve segment outside the cone keeps the sample valid but now Lemma 2.5 is violated which is a contradiction. □

Lemma 3.3. *Consider an ε-sample with $\varepsilon \leq 1/2$. Let p, q, r be a chain of three adjacent samples and let s be a sample not adjacent to q. If p and r do not lie inside the ball B with diameter $\overline{qs}$ there will be a medial axis point on the segment $\overline{qs}$ with distance at most $3/4\|s-q\|$ from q.*

Proof. Figure 3.5 illustrates this and the following lemma. Since p and r are not inside B and q and s are not directly connected, the ball B contains or touches at least two connected components of the curve. Hence a medial axis point must lie on $\overline{qs}$ due to Lemma 2.3. Let m be the one such point closest to q.

Let without loss of generality $\|s-q\| = 1$. Assume m is farther away from q than $3/4$. Then the radius of the empty ball around m which touches the curve is less than $1/4$. Call the touching point adjacent to q x and add it to the sample. This is an allowed operation because we want to prove something about the medial axis which solely depends on the curve. Such a point x must exist because m was chosen to be the medial axis point on $\overline{qs}$ closest to q and otherwise the curve had to leave the cone defined in Lemma 3.2 which would violate Lemma 3.2.

Now there is a point u on the curve between q and x with $\|u-x\| = \|q-x\|/2$ and $\|u-q\| \geq \|q-x\|/2$. The local feature size of u is at most $\|u-x\| + \|x-m\|$ by the triangle inequality. For an ε-sample $\|u-x\| \leq \varepsilon\|u-m\|$ must hold due to Lemma 2.4. Using the above, we get

$$\frac{\|q-x\|}{2} \leq \varepsilon\left(\|u-x\| + \|x-m\|\right) = \varepsilon\left(\frac{\|q-x\|}{2} + \|x-m\|\right). \tag{3.1}$$

Since $\|q-m\| > 3/4$ and the radius of the medial ball around m is at most $1/4$, we have $\|q-x\| \geq 1/2$. Plugging this into Inequality (3.1) together with $\varepsilon = 1/2$ and $\|x-m\| < 1/4$ yields a contradiction. □

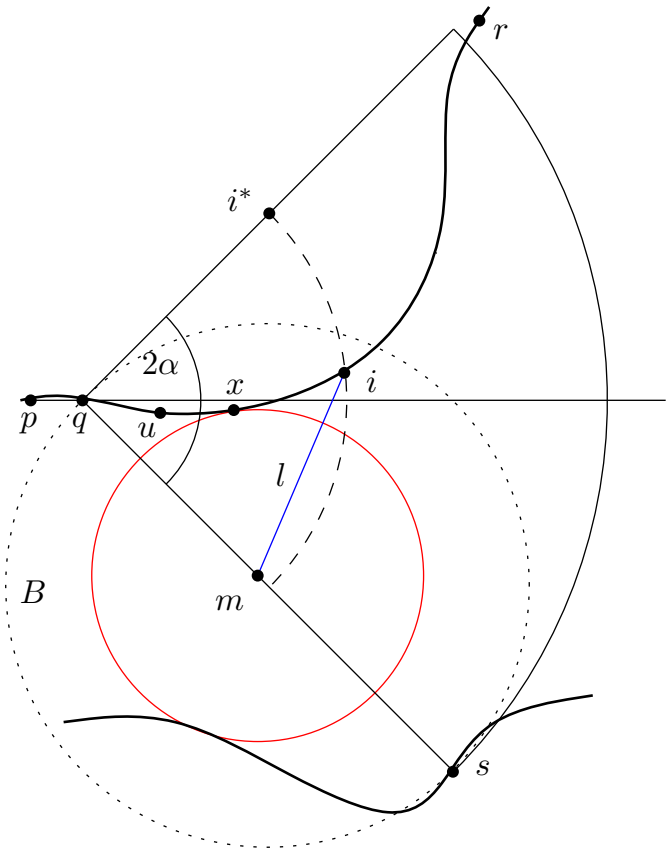


Fig. 3.5: Computing the worst case distance between a point on the curve and the medial axis.

Lemma 3.3 is a modification of Lemma 18 by Althaus [Alt01] which now allows to finish the correctness proof.

Lemma 3.4. *Given an ε-sample of a smooth closed curve with $\alpha > 0$ and $\varepsilon < (13/4 - 3\cos 2\alpha)^{-1/2} < 1/2$ and a correct edge (p, q). The edge (q, s) is correct if and only if s is the closest point to q inside the cone with apex q aligned to (p, q) with opening angle 2α.*

Proof. Assume point s to be the closest to q inside the cone but the edge (q, s) is not correct. Then there is a correct edge (q, r). Due to Lemma 3.2 the curve intersects the dashed circular arc around q with radius $\|q - s\|/2$ in a point i which has at least distance $\|q-s\|/2$ to its nearest sample points q and r, see Figure 3.5.

On the other hand we know by Lemma 3.3 that a medial axis point m is on $\overline{qs}$ and in the worst case it is as far away from i as possible—in $q+3/4\vec{qs}$. So the local feature size of i is at most $l = \|i-m\|$ while the distance to the closest sample point is at least $\|q-s\|/2$ due to Lemma 2.4 and Lemma 2.5. In the worst-case i is at position i^*. For a contradiction we bound ε by $\|q - s\|/(2l)$ and apply the law of cosines to get

$$\varepsilon < \frac{\|q-s\|}{2\|i^*-m\|} = \frac{\|q-s\|}{2\|q-s\|\sqrt{\left(\frac{1}{2}\right)^2 + \left(\frac{3}{4}\right)^2 - 2\frac{1}{2}\frac{3}{4}\cos 2\alpha}} = \frac{1}{\sqrt{\frac{13}{4} - 3\cos 2\alpha}}.$$

For such an ε the sample is no longer valid because the point i^* and hence every possible i has a larger distance to its closest sample point than ε times the local feature size.

The other direction follows directly from Lemma 3.2 and the algorithm. □

Corollary 3.5. *Given an ε-sample of a smooth closed curve with $\varepsilon \leq 0.48$ and a correct edge (p, q). The edge (q, s) is correct if and only if s is the closest point to q inside the cone with apex q aligned to (p, q) with opening angle twice* 0.97 *(roughly* 111°*).*

Proof. Maximize ε under the inequalities from Lemma 2.5 and Lemma 3.4:

$$\varepsilon \leq 2\sin\frac{\alpha}{4} \text{ and } \varepsilon < \frac{1}{\sqrt{\frac{13}{4} - 3\cos 2\alpha}}.$$

For $\alpha = 0.97$ we get $\varepsilon \leq 0.48$. □

The result from these lemmas is now stated as the following theorem.

Theorem 3.6. *Algorithm 1 correctly reconstructs simple, smooth, closed curves from ε-samples using α-probes with $\alpha \leq \pi/2$ and $\varepsilon < \min\{2\sin\alpha/4, (13/4 - 3\cos 2\alpha)^{-1/2}\}$. In particular it reconstructs correctly from* 0.4*-samples using π/2-probes, and from* 0.48*-samples using* 0.97*-probes.*

Proof. Choose the shortest edge as seed edge. This is a correct edge by Lemma 3.1. Using Corollary 3.5 repeatedly shows that Algorithm 1 adds correct edges to the connected component of the seed edge. Since the reconstruction Γ of a simple, smooth, closed curve has exactly one connected component and every vertex has degree two, Algorithm 1 constructs Γ. □

Altogether this guarantees correct results for a whole family of algorithms in any dimension, with different angles $0 < \alpha \leq \pi/2$, and for $\varepsilon \leq 0.48$ in the best case. This is a significant improvement over the original NN-Crust algorithm from 1999 [DK99]. The NN-Crust algorithm is a special case of this approach for $\alpha = \pi/2$. From our more careful analysis, a direct increase of the ε-bound follows for the original algorithm from 1/3 to 0.4.

4. RECONSTRUCTING INTERSECTIONS

4.1 The Idea and Main Result

We have seen in the prior chapter that the new approach can handle smooth, closed curves. In this chapter we show how to use the same algorithm for curves with self-intersections.

medial axis

An ε-sample does not exist for a self-intersecting curve. The *medial axis* goes through the intersection point, hence a infinitely dense sample is required which contradicts Definition 2.4. Therefore we need to modify the original curve to be able to apply *ε-sampling*. The key idea is the removal of small disks around intersections from the original curve. The resulting set is a collection of several smooth, open curves without intersections. For such a set a usual ε-sample exists. Nevertheless, the reconstruction should resemble the original curve including the intersections. The whole process is illustrated in Figure 4.1.

ε-sample

The major problem discussed in the next section is the exact computation of the radii of an exclusion disk. An exclusion disk has an inner radius which circumscribes the part that is actually excluded. The outer radius is needed for the correctness proof to guarantee "niceness" of the curve in the vicinity of the excluded part. Since there are many degrees of freedom and even the probe can be a nearly arbitrary function, it is very hard to exploit general geometric properties. Some of the proofs in this chapter contain numeric bounds resulting from basic calculus. To make these numeric bounds simple to compute, we fix the probe to the simple 0.277-probe defined by $\theta(\beta) = 1 - 3\beta/2$, shown in Figure 4.2. The proofs similarly work for other probes but the bounds have to be worked out individually for each probe.

The main result from this chapter is the following theorem about the reconstructability of curves with intersections.

Theorem 4.1. *Let Σ be a smooth, closed curve with a finite number of self-intersections but no multiple intersections. The maximum curvature of Σ is denoted by c. Let S be an* 0.138*-sample of Σ minus an exclusion disk for every intersection. The exclusion disk for intersection i, with χ being the smaller angle formed by the tangents on Σ in i, has an outer radius r_o and an inner radius r_i, such that*

1. $r_o \leq 2\sin(\chi/86)/c$,

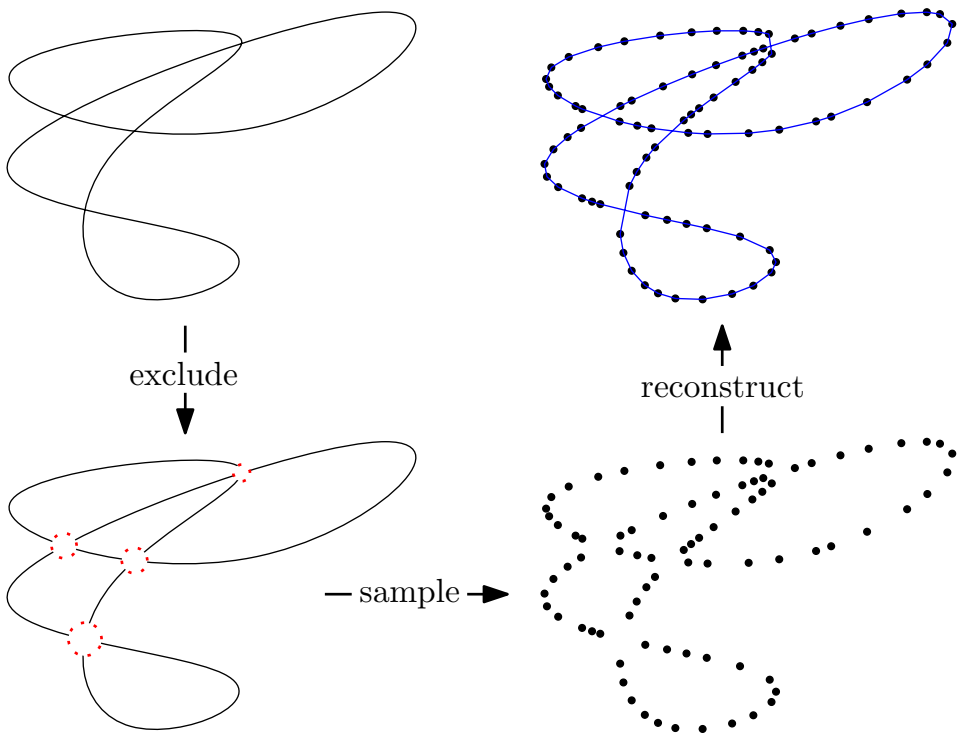


Fig. 4.1: The idea of the exclusion disks: Given a closed curve (top left), exclude circular regions around intersections (bottom left), sample the resulting set of non-intersecting curves (bottom right) and reconstruct it including the intersections from the original (top right).

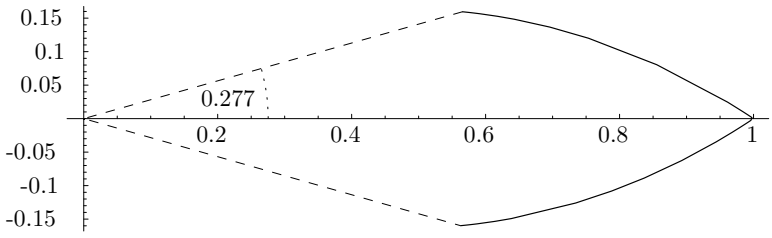


Fig. 4.2: The 0.277-probe for the correctness proof: $\theta(\beta) = 1 - 3\beta/2$.

2. *the ball with radius r_o around i contains exactly one intersection and exactly two components which intersect in i, and*

3. $r_i \leq r_o/6$.

Then Algorithm 1 reconstructs Σ from S.

Proof. The correctness follows from Lemma 3.1 for the seed edge, from Lemma 4.7 for crossing edges and from Lemma 4.9 for all other edges. □

The lemmas 4.7 and 4.9 referred to in the proof are proven in the next sections.

4.2 Computing the Exclusion Disk

4.2.1 Overview

This section provides a global overview of the exclusion disk computation while the next section will describe the details.

To compute the actual exclusion disk centered at a self-intersection point i, we start with a surrounding circle with larger radius also centered at i. The radius r_o of this outer circle is chosen first and has two functions. It must be small enough to isolate the intersection from other intersections and by choosing r_o small enough we can guarantee a certain "flatness" of the curve inside the outer circle and hence inside the exclusion disk. The flatness follows directly from the bounded curvature of Σ, because the curve is smooth. So r_o controls the maximum *turning angle* of the curve inside the outer circle. Making r_o arbitrarily small makes the intersecting curve segments inside the circle approximate straight line segments arbitrarily well.

turning angle

The inner radius r_i, the radius of the exclusion disk, is chosen relative to r_o. It must be so much smaller than r_o, such that at least two sample points are on each of the four curve parts emanating from i within the outer circle. This is easy to achieve because the medial axis is always nearby in the vicinity of intersections, so the sample density must be very high. Removing the exclusion disk then defines four *wedges* within the circle with radius r_o. They contain at least two samples each. The most complicated part now is to consider all possible positions of the samples inside these wedges. One has to show that aligning the probe along two samples in one wedge leads to a sample in the opposite wedge and not to one in the other two wedges or outside of the circle with radius r_o. Figure 4.3 is a sketch of the described situation. To guarantee this last property, the intersecting curve segments must be very "flat", so one has to chose r_o small enough in the initial step.

wedge

Keeping these basic ideas in mind should help not getting lost in the following section containing the actual computation of the mentioned radii.

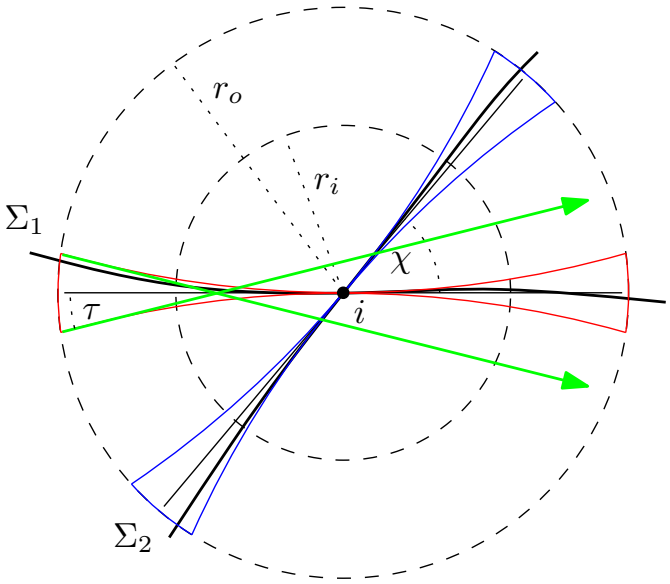


Fig. 4.3: Graphic overview of an exclusion disk an the quantities involved.

Σ_1, Σ_2	The intersecting parts of the curve.
i	The intersection point of Σ_1 and Σ_2.
χ	The smaller angle formed by the tangents of Σ_1 and Σ_2 at i.
r_o	The radius of the outer ball centered at i providing flatness.
r_i	The radius of the exclusion disk.
τ	The maximum ("alignment") angle between the tangent of Σ_1 at i and the tangent of Σ_1 at any other point within the outer ball.

Tab. 4.1: This table lists the quantities involved in computing the exclusion disk.

4.2.2 Detailed Computation

Throughout this section, several variables are introduced. Table 4.1 and Figure 4.3 provide a quick overview.

wedge

Definition 4.1. A *wedge* is a region between the exclusion disk with radius r_i and the outer ball with radius r_o, both centered at a point i. It is bounded by two circular arcs a_1, a_2 of curvature c such that a curve with maximum curvature c going through i tangential to a_1 and a_2 must pass through the wedge. Figure 4.4 shows a typical wedge as it is used throughout this chapter.

The Radii

The Outer Radius. We start the computation of the exclusion disk by determining the outer radius r_o. The value should be small enough to guar-

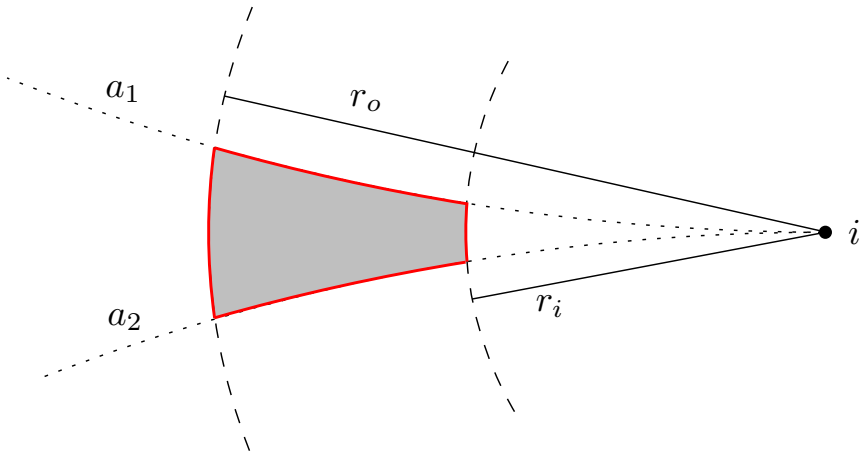


Fig. 4.4: The shaded region with red boundary is a wedge.

antee the required flatness of the intersecting curve segments as described in the previous section. The value depends on the angle χ which is formed by the tangents of the curve parts at the intersection point i. We strive for a maximum turning angle between any possible sample in one wedge p, the intersection point i, and any possible sample in the respectively opposite wedge q of $\sphericalangle(p,i,q) \leq k\chi$ for some fixed $0 < k < 1$. Since we work with smooth curves with bounded curvature at most c, this can be easily achieved by choosing $r_o \leq 2\sin(k\chi/2)/c$ using the following lemma.

Lemma 4.2. *Given a curve Σ with maximum curvature c. The turning angle $\sphericalangle(p,i,q)$ for any three consecutive points $p,i,q \in \Sigma$ with $\|p-i\| \leq r_o$ and $\|q-i\| \leq r_o$ is bounded by β for $r_o \leq 2\sin(\beta/2)/c$.*

Proof. Consider the osculating circle in i. The curvature along the whole curve is bounded by a constant c. The curvature in i is the reciprocal radius of the osculating circle. Hence we can shrink the osculating circle o to radius $1/c$ and get the situation as in Figure 4.5. The point p^* and q^* are the extreme positions for p and q respectively. For any point q between i and q^* holds that the two radii of o ending in i and q are perpendicular to the tangents at these points, hence by simple geometry we get for $\gamma = \angle(q,x,i)$

$$\sin\frac{\gamma}{2} = \frac{\frac{\|i-q\|}{2}}{\frac{1}{c}} = \|i-q\|\frac{c}{2} \leq r_o\frac{c}{2} \leq 2\frac{\sin\frac{\beta}{2}}{c}\frac{c}{2} \leq \sin\frac{\beta}{2}$$

which leads to $\gamma \leq \beta$. So the angle between the tangent at i and the tangent at q is $\gamma/2 \leq \beta/2$. The same holds for p and i by symmetry, hence we have for the total turning angle $\sphericalangle(p,i,q) \leq \beta$. □

If the circle around i with radius r_o contains more than one intersection

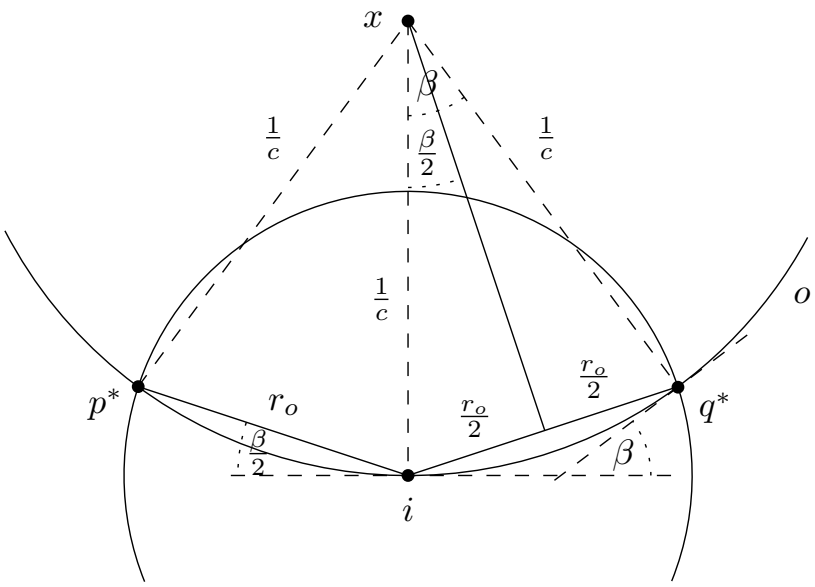


Fig. 4.5: Computation of the outer radius using the osculating circle o with radius $1/c$ at the intersection point i.

or more than one connected component, one must choose a smaller value for r_o such that this is not the case.

The Inner Radius. The inner radius r_i must ensure two things. It has to be so much smaller than r_o, such that at least two samples must be placed in every *wedge*. This limits the possible alignments for the probe. Additionally r_i must be small enough, such that it is guaranteed that an aligned probe hits a sample inside a wedge before any part of the probe leaves the ball with radius r_o.

wedge

The space needed at least for two samples to appear in an ε-sample depends on the local feature size of the points on the particular curve segments.

Lemma 4.3. *Let i be a self-intersection of the curve Σ with maximum curvature c. Let χ be the smaller angle formed by the tangents of Σ at i. Consider a ball B_o with radius r_o, and a ball B_i with radius $r_i < r_o$, both centered at i. For every point $x \in B_o \cap (\Sigma \setminus B_i)$ we have lfs$(x) \leq r_o$. Therefore every wedge contains at least two samples in an ε-sample with $\varepsilon < 1/4$ if $r_i \leq r_o(1 - 4\varepsilon)$.*

Proof. The point i is a medial axis point in $\Sigma \setminus B_i$ because a medial ball around i with radius r_i exists by construction. Therefore the local feature size of any point inside B_o is at most r_o.

Figure 4.6 shows how to bound the distance needed between r_o and r_i. The curve points x, y, z have distance at most $\varepsilon\text{lfs}(x), \varepsilon\text{lfs}(y)$ and $\varepsilon\text{lfs}(z)$ to

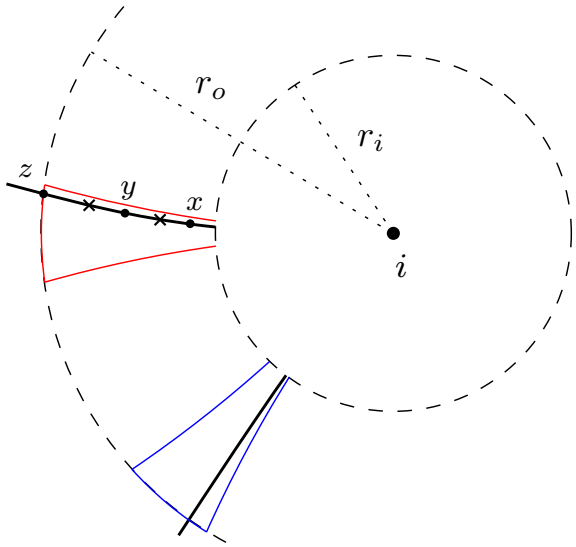


Fig. 4.6: Three points x, y, z on the curve specify the minimal difference between the outer and the inner radius by their local feature sizes.

their closest sample respectively, so the total distance can be bounded by $r_o - r_i \geq 4\varepsilon r_o$. This is equivalent to $r_i \leq r_o(1 - 4\varepsilon)$ and since r_i should be smaller than r_o, it is only valid for $\varepsilon < 1/4$. □

For the guarantee, that an aligned probe does not leave the outer ball before it hits a point, we need the following considerations about possible alignments.

Lemma 4.4. *Given a smooth, closed curve Σ with maximum curvature c, and a point $i \in \Sigma$. Denote the tangent of Σ at i by t. The smaller angle between t and the secant through any two points $p, q \in \Sigma$ with $\|p-i\|, \|q-i\| \leq 2\sin(\tau/2)/c, \tau \leq \pi/2$ is bounded by τ.*

Proof. By the mean value theorem it suffices to look at the tangents of Σ instead of the secants. Obviously the "steepest tangent" is obtained for Σ being a circular arc with curvature c. From Lemma 4.2 and Figure 4.5 we directly get our result. □

Figure 4.7 shows the angle τ computed in Lemma 4.4 on an exclusion disk construction. This angle will be called *alignment angle* in the following paragraphs.

alignment angle

Lemma 4.5. *Let i be a self-intersection of the curve Σ with maximum curvature c. Let χ be the smaller angle formed by the tangents on the curve at i. Consider an exclusion disk around i with outer radius $r_o \leq 2\sin(\tau/2)/c$ for some fixed τ, and inner radius $r_i < r_o$. In every wedge the sample closest to i has distance at most $r_i(1 + \varepsilon\sin((\chi + \tau)/2))$ from i for an ε-sample.*

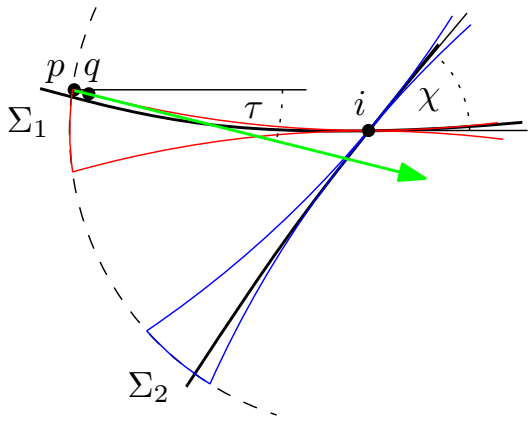


Fig. 4.7: The tangent angle τ in p is the maximum possible deviation from the x-axis with respect to the turning angle.

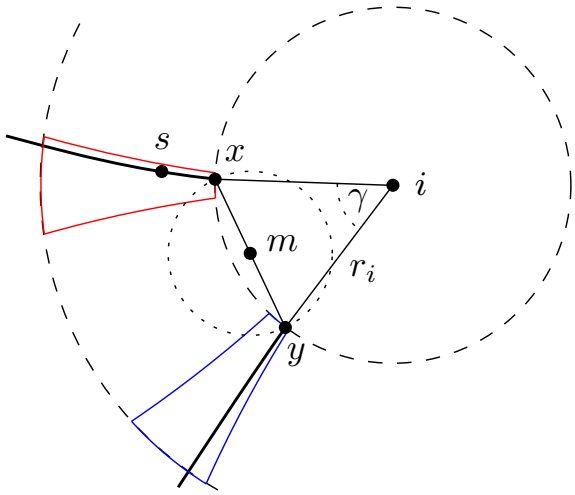


Fig. 4.8: For an ε-sample the distance between the intersection point and its closest sample is bounded by a nearby medial axis point.

Proof. Figure 4.8 illustrates the following construction. Let s be the sample closest to i in one wedge, and let x, y be the curve points on the boundary of the inner ball in the same wedge and the nearby wedge respectively. The ball with diameter $\overline{xy}$ is a medial ball by Definition 2.5 because the curve touches it in x and y and cannot enter it due to the bounded curvature c. Therefore the center is a medial axis point, denoted by m.

Consider the triangle spanned by i, x, y. We have

$$\sin\frac{\gamma}{2} = \frac{\|x-m\|}{r_i}$$
$$\operatorname{lfs}(x) \leq \|x-m\| = r_i \sin\frac{\gamma}{2}$$

and because $\gamma \leq \chi + \tau$ by Lemma 4.2

$$\|x-m\| \leq r_i \sin\frac{\chi+\tau}{2}.$$

By the triangle inequality we get

$$\|s-i\| \leq \|s-x\| + \|x-i\| \leq \varepsilon \operatorname{lfs}(x) + r_i \leq r_i\left(1+\varepsilon\sin\frac{\chi+\tau}{2}\right).$$

□

Lemma 4.6. *Given a smooth, closed curve Σ with maximum curvature c, and a point $i \in \Sigma$. Consider an exclusion disk around i with outer radius $r_o \leq 2\sin(\tau/2)/c$ for some fixed $\tau \leq 0.184$, and inner radius $r_i \leq r_o/6$. In an ε-sample of Σ, $\varepsilon \leq 0.138$, exist three consecutive samples points p, q, r inside the ball with radius r_o around i, such that the probe distance function D_{pq} for the 0.277-probe $\theta(\beta) = 1 - 3\beta/2$ is smaller for r than for any point outside of the ball with radius r_o.*

Proof. We will show two things: the existence of a sample r with finite probe distance, and that all points outside of the ball with radius r_o have larger probe distance than r.

We denote the tangent at i by t. For $r_i \leq r_o/6$ and $\varepsilon \leq 0.138$ we have $r_i \leq r_o(1-4\varepsilon)$ and therefore at least two samples in each wedge by Lemma 4.3. Let p, q be in the same wedge and r in the opposite wedge. By Lemma 4.4 we know that the angle between $\overline{pq}$ and t is at most τ. The angle between $\overline{ir}$ and t is at most $\tau/2$ by Lemma 4.2. Therefore $\sphericalangle(p,q,r) \leq \tau + \tau/2$. To reach r with the 0.277-probe we need $\sphericalangle(p,q,r) \leq 0.277$, which is guaranteed by $\tau \leq 0.184 < 0.277/1.5$. Hence the required point r exists.

The distance between q and r is bounded from above by Lemma 4.5 by $\|q-r\| \leq 2r_i(1+\varepsilon)$, so we have

$$D_{pq}(r) \leq \frac{2r_i(1+\varepsilon)}{\theta(0.277)} < 3.43r_i(1+\varepsilon).$$

For any point s with distance larger than r_o to i we have

$$D_{pq}(s) \geq \frac{r_o - r_i(1+\varepsilon)}{\theta(0)} = r_o - r_i(1+\varepsilon).$$

It remains to show that $3.43r_i(1+\varepsilon) < r_o - r_i(1+\varepsilon)$ which is equivalent to

$$r_i < \frac{r_o}{4.43(1+\varepsilon)}$$

which is true because $r_i \leq r_o/6$ and $\varepsilon \leq 0.138$ by assumption. □

Combining the conditions for both radii results in the following lemma.

Lemma 4.7. *Given two intersecting, smooth curve parts with maximum curvature c. The angle formed by the tangents at the intersection i is called* χ*. The radius* $r_o \leq 2\sin(\tau/2)/c$ *for some fixed* $\tau \leq 0.184$ *of a ball around i is chosen small enough, such that it does not contain additional curve parts or intersections other than i. Remove a disk of radius* $r_i \leq r_o/6$ *and take an* 0.138*-sample S. Then the intersection becomes reconstructed correctly from S by Algorithm 1 using the* 0.277*-probe* $\theta(\beta) = 1 - 3\beta/2$*.*

The extensive proof for this lemma is given in the next paragraphs.

Proof of Lemma 4.7

The Unreachable Wedge. Let Σ_1, Σ_2 be the intersecting curve parts. They both define two wedges each. Next we will show, that it is impossible to start the probe in one wedge of Σ_1 and end in the closer wedge of Σ_2. This is due to the necessary *turning angle* which was already bounded in Lemma 4.2 with the details as follows.

turning angle

Let p be the sample in one wedge of Σ_1 which is closest to the intersection point i, and let w be a sample in the closer wedge of Σ_2. Consider the triangle spanned by p, w, i illustrated in Figure 4.9. The distance from p to i is bounded by the medial axis, so $\|p-i\| \leq r_i\,(1+\varepsilon\sin((\chi+\tau)/2))$ due to Lemma 4.5.

The angle in w is $\pi - \beta - \gamma$ and by the law of sines we have

$$\frac{\sin\beta}{r_i} = \frac{\sin(\pi-\beta-\gamma)}{\|p-i\|} = \frac{\cos\beta\sin\gamma + \sin\beta\cos\gamma}{\|p-i\|}$$

which becomes

$$\beta = \arctan\frac{r_i\sin\gamma}{\|p-i\| - r_i\cos\gamma} \geq \arctan\frac{\sin\gamma}{1+\varepsilon\sin\frac{\chi+\tau}{2} - \cos\gamma}$$

and because $\gamma \geq \chi - \tau$, we get for $\tau \leq \chi/3$

$$\geq \arctan\frac{\sin\gamma}{1+\varepsilon\sin\gamma - \cos\gamma} = \arctan\frac{1}{\varepsilon + \frac{1}{\sin\gamma} - \cot\gamma}.$$

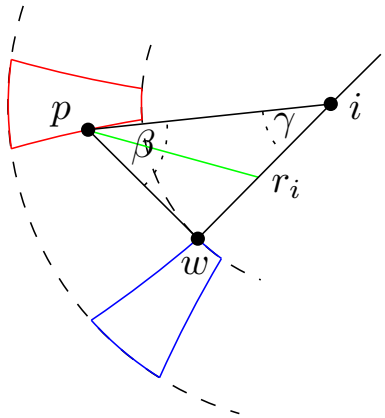


Fig. 4.9: Computing the turning angle necessary to reach a wrong sample w starting from p.

The angle between the tangent at i and $\overline{pi}$ is at most $\tau/2$ and the maximum alignment angle is τ. To show that the wrong wedge cannot be touched, we will show $\beta - 3\tau/2 > 0.277$.

The value for β is monotone decreasing in γ. For $\tau \leq \chi/7$, γ is at most $8\chi/7$, so $\gamma \leq 8\pi/14$ and therefore $\beta > 0.622$. For τ we have $3\tau/2 \leq 3\chi/14 \leq 3\pi/28 < 0.337$. Together we get $\beta - 3\tau/2 \geq 0.622 - 0.337 = 0.285$ which is larger than 0.277, so the probe could never reach a point in this wrong wedge because it would have to turn too much.

The Far Wedge. It remains to show that every sample in the reachable wrong wedge has a larger distance measured with the *probe distance function* than the correct sample. Figure 4.10 shows possible worst-case positions. The samples p, q are the last two in the starting wedge. They control the probe's alignment. The point w is any possible sample in the wrong wedge and r is the worst possible location for a sample in the correct wedge.

probe distance function

Correctness can be expressed formally as

$$\begin{aligned} & \frac{\|q-r\|}{\theta(\sphericalangle(p,q,r))} < \frac{\|q-w\|}{\theta(\sphericalangle(p,q,w))} \\ \Leftrightarrow \quad & 0 < \|q-w\|\theta(\sphericalangle(p,q,r)) - \|q-r\|\theta(\sphericalangle(p,q,w)) \end{aligned} \tag{4.1}$$

The two nominators and denominators can be bounded individually from the proper sides as follows.

The distance between the starting sample q and the correct sample r can be at most twice the maximal distance of an innermost sample in a wedge to i by triangle inequality. The maximal distance of a sample to i depends on the local feature size and is bounded in Lemma 4.5, so we get

$$\|q-r\| \leq 2r_i \left(1 + \varepsilon \sin \frac{\chi+\tau}{2}\right).$$

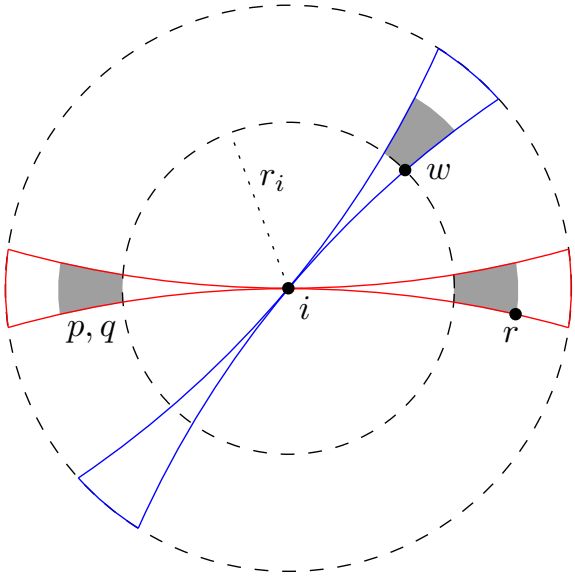


Fig. 4.10: Possible worst-case positions for the samples p, q, r, w. We have to show, that the correct sample r has a smaller probe distance to q than w for all possible probe alignments.

The smallest value for $\theta(\sphericalangle(p,q,r))$ is reached for the largest possible angle $\sphericalangle(p,q,r)$. This is composed of the largest alignment angle τ and the maximum turn to reach r. In total this can be at most $3\tau/2$, so we have

$$\theta(\sphericalangle(p,q,r)) \geq 1 - 3/2(3\tau/2) = 1 - 9\tau/4.$$

The distance between q and w becomes as small as possible, if the angle $\angle(q,i,w)$ is as small as possible. This is similar to the above $\pi - \chi - 3\tau/2$. The distance is then computed using the law of cosines as

$$\begin{aligned} \|q - w\| &\geq \sqrt{2r_i^2 - 2r_i^2 \cos(\pi - \chi - 3\tau/2)} \\ &= r_i\sqrt{2 + 2\cos(\chi + 3\tau/2)} \\ &= 2r_i \cos(\chi/2 + 3\tau/4). \end{aligned}$$

Maximizing $\theta(\sphericalangle(p,q,w))$ means minimizing $\sphericalangle(p,q,w)$ which can be bounded from below as shown in Figure 4.11. With $i = (0,0), q = (x_q, y_q), w =$

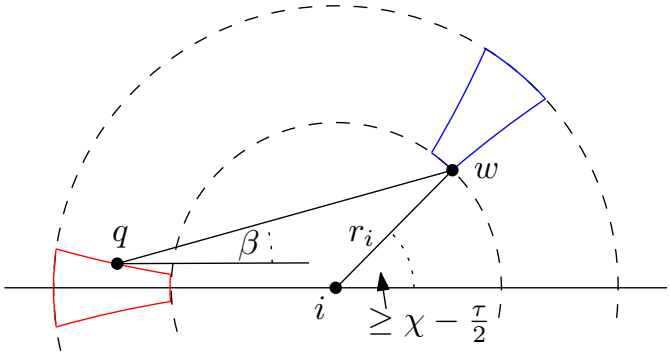


Fig. 4.11: A broad lower bound for the turning angle to the wrong sample w. The angle is at least $\beta - \tau$.

(x_w, y_w) we get

$$\begin{aligned} y_w &\geq r_i \sin\left(\chi - \frac{\tau}{2}\right) \\ y_q &\leq r_i \left(1 + \varepsilon \ \sin\frac{\chi+\tau}{2}\right) \sin\frac{\tau}{2} \\ x_w - x_q &\leq 2r_i \left(1 + \varepsilon \ \sin\frac{\chi+\tau}{2}\right) \\ \frac{y_w - y_q}{x_w - x_q} &\geq \frac{\sin\left(\chi - \frac{\tau}{2}\right) - \left(1 + \varepsilon \ \sin\frac{\chi+\tau}{2}\right)\sin\frac{\tau}{2}}{2\left(1 + \varepsilon \ \sin\frac{\chi+\tau}{2}\right)} \end{aligned}$$

and with $\varepsilon \leq 0.138$ we can bound this by

$$\geq \frac{\sin\left(\chi - \frac{\tau}{2}\right)}{2.276} - \frac{\sin\frac{\tau}{2}}{2} \geq \frac{\sin\left(\chi - \frac{\tau}{2}\right)}{2.276} - \frac{\tau}{4}.$$

This last bound is a concave function because it is dominated by the sine expression. We claim that for $\tau \leq \chi/43$ and $0 \leq \chi \leq \pi/2$ it is not smaller than $\chi/4$. It suffices to check the extreme values. For $\chi = 0$ we get $0 \geq 0$ and for $\chi = \pi/2$ we get $\sin(85\pi/172)/2.276 - \chi/172 > 0.430 > \pi/8$. In total it is

$$\sphericalangle(p, q, w) \geq \beta - \tau = \arctan\frac{y_w - y_q}{x_w - x_q} - \tau > \arctan\frac{\chi}{4} - \tau.$$

Plugging all this into Inequality (4.1) with $\tau \leq \chi/43$ yields

$$\begin{aligned} & 0 < 2r_i \cos\frac{89\chi}{172}\left(1 - \frac{9}{172}\chi\right) - 2r_i\left(1 + \varepsilon \sin\frac{44\chi}{86}\right)\theta(\sphericalangle(p,q,w)) \\ \Leftrightarrow \quad & 0 < \cos\frac{89\chi}{172}\left(1 - \frac{9}{172}\chi\right) - \left(1 + \varepsilon\sin\frac{44\chi}{86}\right)\left(1 - \frac{3}{2}\left(\arctan\frac{\chi}{4} - \frac{\chi}{43}\right)\right). \end{aligned}$$

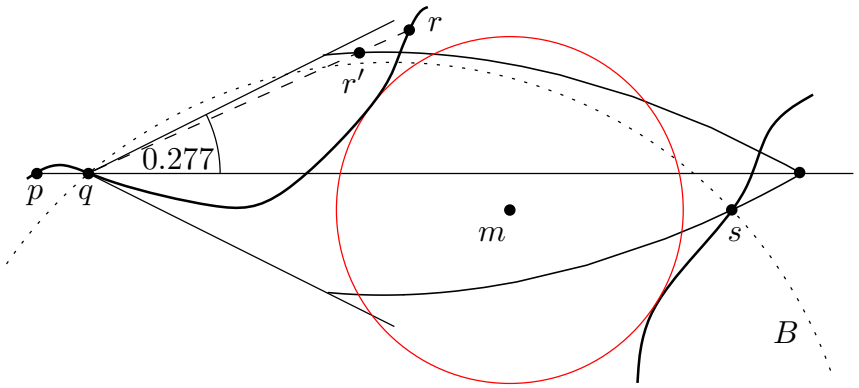


Fig. 4.12: Computing the necessary ε to guarantee correct reconstructions for a special non-circular probe.

We call the right-hand side $f(\chi)$. To prove that $f(\chi)$ is strictly positive for $\chi > 0$, we will find a lower bound for f which is a concave function, and then we evaluate the extreme values.

$$\begin{aligned} f(\chi) &= \cos\frac{89\chi}{172}\left(1-\frac{9}{172}\chi\right) - \left(1+\varepsilon\sin\frac{44\chi}{86}\right)\left(1-\frac{3}{2}\left(\arctan\frac{\chi}{4}-\frac{\chi}{43}\right)\right) \\ &\geq \cos\frac{89\chi}{172} - \frac{9}{172}\chi - 1 - \varepsilon\sin\frac{44\chi}{86} + \frac{3}{2}\left(\arctan\frac{\chi}{4}-\frac{\chi}{43}\right) \\ &\geq \cos\frac{89\chi}{172} - \frac{9}{172}\chi - 1 - \varepsilon\frac{44\chi}{86} + \frac{3}{2}\arctan\frac{\chi}{4} - \frac{3\chi}{86} =: g(\chi) \end{aligned}$$

This bound $g(\chi)$ is the sum of linear and concave functions, therefore it is concave. The function values at the boundary are $g(0) = 0$ and $g(\pi/2) > 0.0008$. It follows that $g(0) = f(0)$ is the only root of g and f in the interval $[0, \pi/2]$ because by the definition of concave functions $g(\lambda\pi/2) \geq \lambda g(\pi/2)$ for every $0 \leq \lambda \leq 1$.

This shows that Inequality (4.1) holds. Therefore every possible sample in the correct wedge has a smaller probe distance than any sample in the far wrong wedge. This concludes the proof of Lemma 4.7.

4.3 Correctness for Curve Segments between Intersections

The correctness proof for curve pieces which are not in the vicinity of intersections is comparable to the one from Section 3.3.2 for the NN-Crust algorithm. The important difference is the change in the probe's shape. It is no longer a circular arc, so it allows a wrong point s to minimize D_{pq} although the euclidean distance $\|q-s\|$ is by a factor up to $\theta(\sphericalangle(p,q,s))/\theta(\sphericalangle(p,q,r))$ larger than the distance from q to the correct point r, see Figure 4.12.

The following lemmas are variations of Lemma 3.3 and Lemma 3.4 from Section 3.3.2, taking the different probe shape into account. The results are

weaker than the ones from the former lemmas due to the more complicated non-circular probe shape.

Lemma 4.8. *Consider an ε-sample with $\varepsilon \leq 0.138$. Let p, q, r be a chain of three adjacent samples, and s be a sample not adjacent to q. If $D_{pq}(s) < D_{pq}(r)$, a medial axis point m exists with distance to q at most $1.823 D_{pq}(s)$.*

Proof. We are going to construct a ball B which contains q and w and does not contain p and r. Since p and r are not inside B and q and w are not directly connected on the curve, the ball B contains or touches two connected components of the curve. Hence B contains a medial axis point by Lemma 2.2. It remains to show that such a B exists, and to bound its diameter.

Since $D_{pq}(s) < D_{pq}(r)$ a point r' on the line segment $\overline{qr}$ exists such that $D_{pq}(r') = D_{pq}(s)$. Without loss of generality, r' is left of the oriented line $\overline{pq}$, as in Figure 4.12. We call this the "upper half" of the probe during this proof. Without loss of generality, we assume q has the coordinates $(0, 0)$ and the tip of the probe has coordinates $(1, 0)$. This implies $D_{pq}(s) = 1$. For $\|q - s\| \leq \|q - r\|$ we can take the ball with diameter $\overline{qs}$, and we are done. Otherwise, we have $\|q - s\| > \|q - r\| > \|q - r'\|$.

Lemma 2.5 restricts the turning angles from $\overline{pq}$ to points on the upper half of the probe to the interval $[0, 0.277]$ for $\varepsilon \leq 0.138$ and our 0.277-probe. In this interval, sin and cos are monotone functions.

We will show that the ball B which has its center (x_c, y_c) on the line $y_c = -3/4$ and passes through q and s has the desired property. For (x_c, y_c) and the diameter d of B we have $x_c^2 + y_c^2 = d^2/4$ because $q = (0, 0)$ is on B's boundary.

We start with the case that s and r' are both on the upper half of the probe. The point s has coordinates $(\theta(\beta)\cos\beta, \theta(\beta)\sin\beta)$ for some fixed $\beta \in [0, 0.277]$. We have to construct B such that it does not contain the upper part of the probe left of s. Therefore we have to show that every point of the probe $(\theta(\gamma)\cos\gamma, \theta(\gamma)\sin\gamma)$ has distance at least $d/2$ from (x_c, y_c) for all $\gamma \in [\beta, 0.277]$. So we have

$$
\begin{aligned}
& (\theta(\gamma)\cos\gamma - x_c)^2 + (\theta(\gamma)\sin\gamma - y_c)^2 \geq d^2/4 \\
\Leftrightarrow \quad & x_c^2 + y_c^2 + \theta(\gamma)^2\left(\cos^2\gamma + \sin^2\gamma\right) - 2\theta(\gamma)(x_c\cos\gamma + y_c\sin\gamma) \geq d^2/4 \\
\Leftrightarrow \quad & \theta(\gamma) - 2(x_c\cos\gamma + y_c\sin\gamma) \geq 0 \\
\Leftrightarrow \quad & \frac{\theta(\gamma) - 2y_c\sin\gamma}{2\cos\gamma} \geq x_c. \qquad (4.2)
\end{aligned}
$$

For $\gamma = \beta$, this inequality becomes an equality because s lies on the boundary. We have to show

$$
\frac{\theta(\gamma) - 2y_c\sin\gamma}{2\cos\gamma} \geq \frac{\theta(\beta) - 2y_c\sin\beta}{2\cos\beta}.
$$

Plugging in $\theta(\gamma) = 1 - 3\gamma/2$ yields

$$\frac{1 - 3\gamma/2 - 2y_c \sin\gamma}{\cos\gamma} \geq \frac{1 - 3\beta/2 - 2y_c \sin\beta}{\cos\beta}. \tag{4.3}$$

Consider the following function g and its derivative g'.

$$g(\gamma) = \frac{1 - 3\gamma/2 - 2y_c \sin\gamma}{\cos\gamma}$$
$$g'(\gamma) = \frac{(1 - 3\gamma/2)\sin\gamma - 3/2\cos\gamma - 2y_c}{\cos^2\gamma}$$

In the interval $[0, 0.277]$ we have $(1-3\gamma/2)\sin\alpha - 3/2\cos\gamma - 2y_c \geq 0 - 3/2 - 2y_c$, hence for $y_c = -3/4$ the derivative g' is non-negative and therefore g is monotone increasing in that interval. Inequality (4.3) follows immediately because $\beta < \gamma$.

Now we can bound $x_c < 0.518$ by Inequality (4.2) with $y_c = -3/4$ and get $d^2/4 \leq 0.518^2 + (-3/4)^2$ and hence the diameter of B is less than 1.823.

In the case that s lies on the lower half of the probe, we can simply take the circle through q and the point $(1, 0)$ which contains s and does not contain any point on the upper half of the probe. This is a special case of the above inequalities with $\beta = 0$. All other cases are symmetric to the two discussed ones. □

Lemma 4.9. *Given an ε-sample S of a smooth closed curve with $\varepsilon \leq 0.138$ and a correct edge (p, q). The edge (q, s) is correct if and only if for all points $r \in S \setminus \{q, s\}$ inside the cone with apex q aligned to (p, q) with opening angle 2×0.277, $D_{pq}(s) < D_{pq}(r)$ is true.*

Proof. Assume point s minimizes the function D_{pq} among the points inside the cone but the edge (q, s) is not correct. Then there is a correct edge (q, r). The point r lies outside of the probe but inside the cone, so $\|q - r\|$ is at least $\theta(0.277)D_{pq}(s)$. A point x on the curve between q and r exists with $\|q - x\| = \theta(0.277)D_{pq}(s)/2$ and $\|r - x\| \geq \theta(0.277)D_{pq}(s)/2$.

On the other hand we know by Lemma 4.8 that a medial axis point is at most $1.823D_{pq}(s)$ away from q. So the local feature size of x is at most $\|q-x\|+1.823D_{pq}(s)$ by triangle inequality, while the distance to the closest sample point is $\|q - x\|$. For a valid ε-sample, the following must be true.

$$\begin{aligned}
& \|q - x\| < \varepsilon(\|q - x\| + 1.823D_{pq}(s)) \\
\Leftrightarrow \quad & \theta(0.277)D_{pq}(s)/2 < \varepsilon(\theta(0.277)D_{pq}(s)/2 + 1.823D_{pq}(s)) \\
\Leftrightarrow \quad & \theta(0.277) < \varepsilon(\theta(0.277) + 3.646) \\
\Leftrightarrow \quad & \frac{\theta(0.277)}{\theta(0.277) + 3.646} < \varepsilon \\
\Rightarrow \quad & 0.138 < \varepsilon
\end{aligned}$$

Choosing $\varepsilon \leq 0.138$, the point x has a larger distance to its closest sample than ε times its local feature size and hence the sample is not valid. This contradicts the assumption. □

This completes the correctness proof of Theorem 4.1, page 39.

4.4 Collections of Intersecting Curves

Theorem 4.1 shows that the provided algorithm can reconstruct a single curve with self-intersections. The extension to multiple curves which may self-intersect and intersect each other is trivial. Since the exclusion disks cut a single curve into open pieces, it does not matter, whether these pieces are from one curve or from several curves. The only difference is that for each curve a *seed edge* has to be found. This procedure is already included in Algorithm 1. Therefore collections of intersecting curves are reconstructed as easily as single curves without any changes.

seed edge

5. ANALYSIS OF THE RUNNING TIME AND THE SPACE REQUIREMENTS

5.1 Advanced Data Structures

5.1.1 Dynamic Closest Pair Maintenance

fair-split tree

Bespamyatnikh introduced a lazily updated tree structure, a *fair-split tree*, for closest pair maintenance in 1995 [Bes95]. Points are not separated by their median like in a kd-tree for example but they are divided "fairly" such that at least a fixed fraction of the elements is in the one subtree and the rest is in the other subtree. This roughly balanced tree allows a certain laziness performing updates. The tree only has to become rebalanced if the ratio between two subtrees becomes worse than the allowed fraction. An involved analysis shows that this data structure efficiently maintains the dynamic closest pair of points. The initial tree of size $O(n)$ can be built in $O(n \log n)$ time and each operation (insert point, delete point, query closest pair) takes only $O(\log n)$ time. This approach works for any fixed dimension.

5.1.2 Probe Inflation with Partition Trees

partition tree

A *partition tree* is a data structure which partitions a point set in $\mathbb{R}^d$ into r disjoint subsets. These subsets are partitioned recursively by the same technique. The partitioning is done such that the subsets have roughly equal size and they can be enclosed by triangles such that any query line does not intersect more than $O\big(r^{1-1/d}\big)$ triangles. Therefore a range query counting the number of points inside a polygon with a constant number of edges can be executed in $O\big(n^{1-1/d}\big)$ 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 [Mat93a]. The same author also provides a detailed analysis in [Mat93b].

Partition trees can be used for probe inflation in the following way. For each of the r triangles on one level of the partition tree, a single point (arbitrary or random) is tested whether it lies inside the current 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 triangle must be completely outside or it intersects the probe. The outer triangles are excluded and the algorithm is performed

recursively on the intersected triangles. The analysis is the same as for a normal range query on partition trees.

Although the original analysis uses polygonal query shapes, the described procedure works for arbitrary shapes which allow easy intersection testing.

5.2 Practical Efficiency with kd-Trees

The kd-tree is a very common space partitioning data structure introduced by Bentley [Ben75]. Very efficient implementations exist for nearly every programming language.

probe distance function

In practice both our important queries, nearest unmarked neighbor and point minimizing the *probe distance function*, can be executed very fast using only a single simple kd-tree. At least for two or three dimensions this performs very well for moderately sized point sets compared to the other advanced data structures proposed due to very small runtime constants.

Finding α-neighbors has a very high locality in expectation, so well-known space partitions, e.g. quad-trees or ham-sandwich-trees, will most likely also work well.

For high dimensions the kd-tree performance for random point sets degrades noticeably. Our sample points are obviously not randomly distributed but well chosen instead. Thus we expect the density to be independent of the dimension. This might be an interesting point for future optimizations which is not discussed further in this thesis.

5.3 Runtime

Let n denote the number of points in the ε-sample. 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.

seed edge

The runtime is composed of these two main steps resulting in a total of $O(P + sS + eE)$ steps for s seed edges and e edges in total with P preprocessing time and S, 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 *probe distance function* D, setting variables and adding edges to the reconstruction are accounted for as constant time operations.

probe distance function

A trivial implementation would not use any preprocessing. Finding a seed edge brute-force requires $O\big(n^2\big)$ time and a single consecutive edge is found in $O(n)$ time. This results in a total of $O\big(sn^2 + en\big) \subseteq O\big(n^3\big)$.

Using data structures proposed in the prior section, we get the following theorem.

Theorem 5.1. *Algorithm 1 reconstructs smooth, closed curves in $\mathbb{R}^d$ with intersections from 0.138-samples with exclusion disks in $O\big(n^{2-1/d}\big)$ steps.*

Proof. The correct reconstruction result follows directly from Theorem 4.1. We use the maintenance of the the closest pair by a fair-split tree. It requires $O(\log n)$ time per operation. Every sample which becomes part of the reconstruction is removed from the *fair-split tree*. The preprocessing takes $O(n \log n)$ time, $S \in O(\log n), E \in O(n + \log n)$. The total runtime is reduced to $O(en)$. 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 $O(n)$ to $E \in O\left(n^{1-1/d}\right)$, yielding the final runtime of $O(P + sS + eE) = O\left(en^{1-1/d}\right)$ for e edges. Since every sample has degree two in the reconstruction, the number of edges is exactly the number of samples, $e \in O(n)$. □

fair-split tree

partition tree

5.4 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 $O(\log n)$ per found consecutive edge but looking for this edge using the partition tree already takes $O\left(n^{1-1/d}\right)$ which dominates the term.

This allows a reconstruction with storage size $\Theta(n + |\Gamma|)$ where n is the number of samples in the input set and Γ the size of the output. Since the output has linear size, the total storage used is $O(n)$. This is clearly optimal.

Theorem 5.2. *Algorithm 1 correctly reconstructs smooth, closed curves in* $\mathbb{R}^d$ *with intersections from* 0.138*-samples with exclusion disks in* $O\left(n^{2-1/d}\right)$ *steps using* $O(n)$ *space.*

6. EXPERIMENTAL RESULTS

6.1 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 [KRG]. A screenshot is shown in Figure 6.1.

Some brute-force implementations were done using C++ but they ran as slow as expected since the runtime is cubic in the number of points in the worst case. For tens of thousands of points, the runtime was already in the range of minutes.

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 curves and with equally spaced samples 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.

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 possible further research.

6.2 Heuristic Results

Another series of experiments were used to show the practicability of the algorithm as a heuristic approach. It is not always possible to guarantee the validity of an ε-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 6.2(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 uniformly at random 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 are based 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.

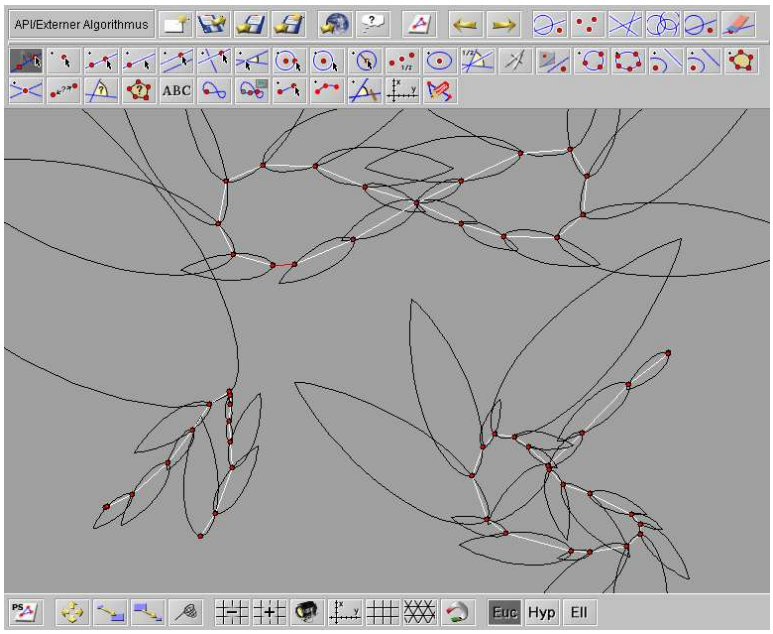


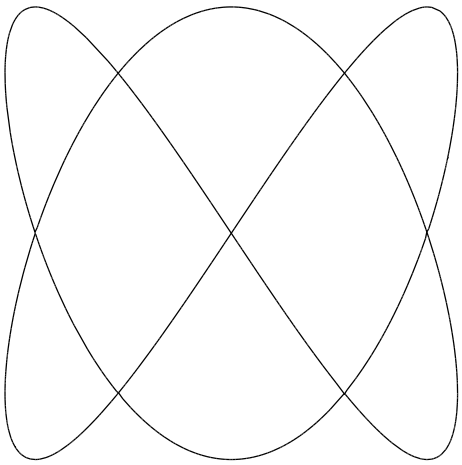
Fig. 6.1: Screenshot from Cinderella with the interactive curve reconstruction plug-in showing the reconstructed curves in white and the inflated probes in black.

Figure 6.2(b) shows the reconstruction for 50 points which is quite bad and definitively undersampled. In Figure 6.2(c) 100 points were taken. Some bends are too sharp to be captured and there are problems caused by wrong connections in the vicinity of intersections. The result for 200 points in Figure 6.2(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 overall appearance 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.

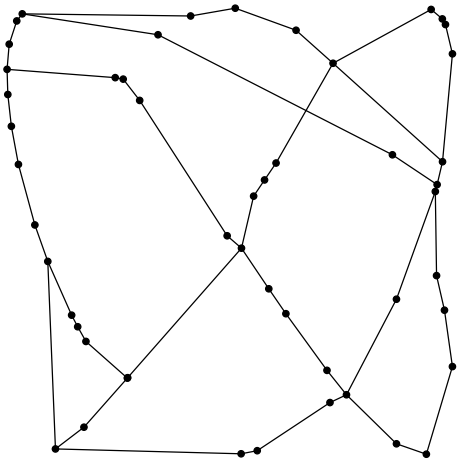
Obviously one can always create a point pattern around an intersection which will result in a wrong reconstruction independent of the density. Nevertheless there are no additional wrong edges besides close to intersections and no gaps, so the reconstructed topology will be correct if the sample is dense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

6.3 Extension to Open Curves

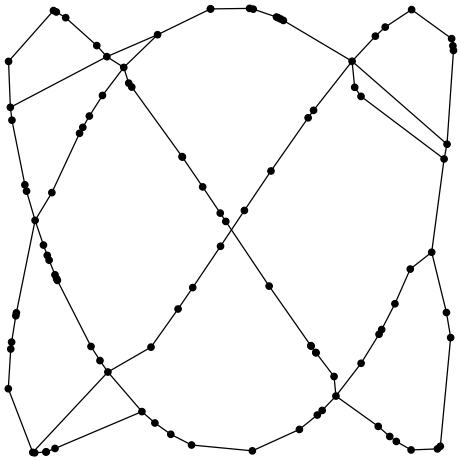
We also experimented with the reconstruction of open curves. The experimental results can be seen in [Len05a]. The idea for open curves is a sample



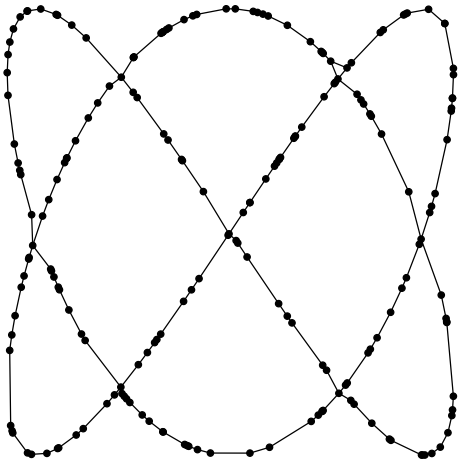
(a) This figure was sampled and reconstructed.



(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.

which is not to dense. The probe must have a negative extent, see Definition 3.1. Then for a probe aligned at the edge (p, q), the point p minimizes the *probe distance function* D_{pq}. This results in an edge (q, p) which is already part of the reconstruction, since the edges in the reconstruction are not oriented. This makes the algorithm stop.

probe distance function

To avoid the reconstruction of wrong endpoints, the ratio between larges sample distance and smallest sample distance must be bounded for samples which are not endpoints. A suitable criterion for this is contained in the (ε, δ)*-sample* or the *uniform* ε*-sample* [Eri03]. It is a refinement of the ε-sample with the additional condition that for any two samples p, q, $\|p-q\| > \delta\text{lfs}(p)$. This approach is by far not as well established as the ε-sample and is therefore not investigated here. Nevertheless it works well as a heuristic and does not require any changes to the very simple algorithm.

(ε, δ)*-sample*
uniform ε*-sample*

7. CONCLUSION

In this thesis a novel technique for curve reconstruction in arbitrary dimension is introduced. A simple and short algorithm is provided which has the guarantees as the Nearest-Neighbor-Crust as a special case but is much more flexible. The minor increase in runtime compared to other curve reconstruction algorithms is outweighed by the fact, that this is the first algorithm which can handle self-intersecting curves and collections of intersecting curves. This is achieved by a small modification of the well-known ε-sampling condition, which would otherwise not be applicable to intersecting curves. Although the correctness proof is fairly involved, it all boils down to one simple radius computation per intersection.

A detailed analysis is provided for several data structures. The practicability of the algorithm as a heuristic without a proper sample is shown in experiments. An interactive applet is available on the Internet.

The major contribution of this thesis is to show a general possibility to reconstruct intersecting curve. This is fully analyzed but still allows a wide variety of specializations and optimizations which might be subject to future research.

Part II

APPROXIMATING THE MEDIAN IN STREAMS WITH CONSTANT STORAGE

ABSTRACT

In this part the well-known problem of finding the median of an ordered set is studied under a very restrictive streaming model with sequential read-only access to the data. Only a constant number of reference objects from the stream can be stored for comparison with subsequent stream elements. A first non-trivial bound of $\Omega(\sqrt{n})$ distance to the extrema of the set is presented for a single pass over streams which do not reveal their total size n. This result is extended to an algorithm which guarantees a distance of $\Omega\left(n^{1-\varepsilon}\right)$ to the extrema. Additional results about upper bounds, multi-pass algorithms, and arbitrary quantiles are presented.

ZUSAMMENFASSUNG

In diesem Abschnitt wird das bekannte Problem der Mediansuche in einer geordneten Menge untersucht. Dies erfolgt in dem sehr strengen Streaming-Modell, welches nur Lesezugriff auf die Daten erlaubt. Gleichzeitig darf nur eine konstante Anzahl von Referenzobjekten aus dem Datenstrom für Vergleiche gespeichert werden. Die erste nicht-triviale untere Schranke von $\Omega(\sqrt{n})$ für den Abstand zwischen dem gefundenen Wert und den Extremwerten des Datenstroms wird mit einem Single-Pass-Algorithmus erreicht. Dabei ist es nicht erforderlich die Länge n des Datenstroms zu kennen. Dieser Algorithmus bildet den Grundbaustein für ein Approximationsverfahren, welches einen Abstand von $\Omega\left(n^{1-\varepsilon}\right)$ zu den Extremwerten garantiert. Des Weiteren werden obere Schranken, Multi-Pass-Algorithmen und Ansätze für andere Elemente als den Median präsentiert.

8. INTRODUCTION

8.1 Motivation

In any finite subset S of a totally ordered set, every element $x \in S$ splits S into a set $S_<$ of elements smaller than x and a set $S_>$ of elements larger than x. The ratio between $|S_<|$ and $|S_>|$ determines how good x *splits* S. The best possible splitter is the *median* which is the element of rank $\lceil n/2 \rceil$. The *rank* of an element is its position in the sorted order of the set S.

splitter
median
rank

Many applications require a good splitter. The crux of every divide-and-conquer algorithm is finding an element which partitions a data set of size n into two parts of roughly equal size. A well-known example is quicksort, which performs best, splitting in each recursion at the median.

We study the problem of finding a good splitter—approximating the median—under very rigorous restrictions. We use the *streaming model*, allowing only a single pass over the data. Furthermore we allow only a constant number of elements to be stored. The presented algorithms are deterministic, providing a guarantee on the quality of the output. This single-pass setting with constant storage has several applications.

streaming model
sensor networks

Sensor networks typically have very small and cheap components, so the amount of available memory on a single sensor is tiny. In a network with several sensors, e.g. measuring the temperature every few seconds, the median and other quantiles are significant quantities. No sensor has enough memory to store the whole data, so the information is broadcast and has to be processed in real-time in the order of reception. This is modeled exactly by our approach with constant memory and no assumptions about the order of the stream elements.

Another class of application are *databases*. Fast estimation of quantiles is important for planning and executing queries, i.e. estimating the size of intermediate results or partitioning the data for parallel query processing. Obviously the estimation should be performed very fast and must happen under memory restrictions due to the size of today's databases compared to the size of main memory in today's computers. Modern servers process thousands of queries per second, so obviously each single process has only very limited resources available.

database

Considering the distribution of the data as unknown, a numerical approximation of the median is not appropriate. The value of such an approximation might be very close to the real median and simultaneously their

positions in the sorted data might be arbitrarily far apart. Depending on the actual problem, a numerical approximation may not be satisfying because in general no element exists with this value. For non-numerical data such an approximation might be very difficult or not useful anyway.

In the classical model of computation, Blum et al. [BFP+73] showed that finding the median is possible in linear time which is optimal. Their algorithm and all known modifications require random access to the data and either modify the original order of the elements or need $\Omega(n)$ additional memory. The past research for exact bounds on the number of comparisons needed and related information was collated by Paterson [Pat96].

8.2 The Streaming Model, the I/O Model, and the Number of Sequential Passes

streaming model

Today's modern communication revived several models of computation originated from storage on tape, namely the *streaming model* and the I/O model. Data which is streamed through a sensor network, the Internet or any other network comes in a strict sequential order forbidding random access and modification. Usually even if random access is possible, sequential or at least locally coherent access would be many times faster due to caching mechanisms. Models appeared, taking the mentioned restrictions into account. The idea of counting and minimizing the number of sequential passes over the data as runtime measurement arose. For most practical purposes, the number of passes is restricted to one or a very small constant.

Typically the total amount of data will not fit into memory, e.g. on a sensor node. Furthermore a data stream might not give away its total size beforehand. The size might even be unknown throughout the whole execution of the algorithm. The stream might just end without premonition. In some cases, a continuing stream has no designated end at all. An approximation algorithm for streams of unknown length must therefore maintain a good estimate at all times and cannot use the data size n in its decisions. We discuss streams both with known and unknown sizes in this paper.

I/O model

A related but more relaxed model is the *I/O model* which is usually applied if the data size exceeds the main memory. In these cases, read/write accesses to blocks of the slow external memory are much more expensive, with respect to runtime, than most algorithmic steps. This again leads to algorithms which prefer local computations on elements which are stored in the same block on the external memory. Allowing random access collides with our application of sequentially transmitted data and is therefore not discussed.

We focus on the streaming model with the following restrictions.

- Sequential access to the data
- Data is read-only

- Storage for a constant number of data elements

The known optimal linear time median algorithms [BFP+73] violate the above conditions and are therefore not applicable in the streaming model.

8.3 Previous Work

8.3.1 The Secretary Problem

Problems where one has to make a decision without total knowledge about the input are quite popular, not only among mathematicians. Choice under uncertainty is a key element in every management seminar as part of the topic *decision theory* [GW04]. Very well-known mathematical problems of that kind are the so-called *secretary problem* and its variations. The first formal appearance of this problem was in 1964 in an article by Chow et al. [CMRS64]. In their formulation, a permutation of a finite subset of a totally ordered universe is presented sequentially. Every element only reveals its *relative rank*, the rank among the elements seen so far. The task is to stop at a certain element x, such that the probability of x being the global maximum is maximized. It is known, that the optimal strategy for n elements skips the first n/e elements, where e is the Eulerian constant. Then it takes the first of the remaining elements which exceeds all of the skipped ones. A useful variant demands that the selected element should be as close to the global maximum as possible. Bruss [Bru05] recently wrote a summary about *Robbins' problem*, which is closely related.

decision theory

secretary problem

relative rank

Robbins' problem

The main ingredient of these problems and their solutions is randomness. Known strategies approximate an extreme element from a sequential stream. In this article, the opposite problem is discussed: how to find an element in the middle, which splits the input sequence into parts of more or less equal size, under uncertainty?

8.3.2 Related Algorithms

Good splitters have high probability. A simple strategy for known n would be to pick a random element with probability $1/n$. This will split the data in a ratio not worse than $1 : 3$ with a probability of 50%. Using the median over a constant sized random sample yields even better results. Vitter overcomes the problem of not knowing n with a special sampling technique called *reservoir sampling* [Vit85].

reservoir sampling

A lower bound of $\Omega(1/\varepsilon)$ space for an ε-approximation algorithm for the median was shown by Rauch Henziger et al. [RRR98]. Their proof utilizes *communication complexity* and reduction to other problems. Both topics are important for fundamental lower bounds in the streaming model but they are not discussed further in this thesis.

communication complexity

Storing only five reference values from the stream, Jain and Chlamtac [JC85] obtain numerically good results but they approximate the value of the median by a special formula. This depends on the distribution and does not yield any guarantees concerning the rank of the returned element.

Granting more than constant memory allows a $(1+\varepsilon)$-approximation for finding the element with rank k. Manku et al. [MRL98] presented a single-pass algorithm with $O\big(1/\varepsilon \log^2 \varepsilon n\big)$ memory if n is known. This was improved to $O(1/\varepsilon \log \varepsilon n)$ memory, not requiring the knowledge of n by Greenwald and Khanna [GK01]. In the former paper, the authors also showed a $(1+\varepsilon)$-approximation technique with probability δ using $O\Big(\frac{1}{\varepsilon} \log^2 \Big(\frac{1}{\varepsilon} \log \frac{1}{1-\delta}\Big)\Big)$ memory.

Munro and Paterson [MP80] studied the problem under several memory restrictions but for constant memory, they only gave a multi-pass algorithm for the exact median without intermediate approximation results after each pass. The authors themselves call it "intolerable in practice". Later Munro and Raman [MR96] solved the problem of finding the median with minimal data movements in $O\big(n^{1+\varepsilon}\big)$ steps with constant extra space but allowed random access to the read-only data.

Very recently Guha and McGregor [GM06] exploited the fact that the distribution of the elements in a stream is not completely adversarial in expectation, although the actual distribution might be unknown. They obtain an element with rank $n/2 \pm O\big(n^{1/2+\varepsilon}\big)$ using polylogarithmic space. Guha et al. [GMV06] analyzed the entropy of streams even further and compared the random order model [MP80] with several oracle models.

These results are only with high probability or require sophisticated structures and more than constant memory. This thesis tackles the problem of finding a simple deterministic algorithm with constant memory which splits a stream with more than a constant number of elements in both parts. Parts of this thesis have already been published in [Len06a].

8.4 The Considered Model

We assume that the stream contains comparable objects with a total order defined on them. For a weak or partial order, the median is not uniquely defined, but with a sensible definition or by embedding the data in a strict total order, the extension of our algorithms should be straightforward. Two cases are analyzed, one where the total number n of elements in the stream is known in advance and in the other case it is not known and cannot be guessed or computed from the stream.

pop
top

We assume the existence of two stream operations: `pop` provides the top element and removes it from the stream and `top` provides the top element without removing it. The operations are intentionally named like the corresponding stack counterparts, because our stream behaves like a filled stack

without the possibility to push objects onto it.

In its local memory, the algorithm cannot store more than a fixed number of references to objects in the stream, later called *markers*. The algorithm can access only the marked elements and the current top element from the stream, so the following interaction with the stream is possible. *marker*

- Compare the markers with each other and with the current top element
- Replace one of the markers by the current top element
- Proceed to the next input

No arithmetic operations with the markers are allowed (like C-style pointer arithmetic) and the return value of the algorithm must be a marker pointing to an element.

Munro and Paterson [MP80] observed that there is no hope for more than a constant distance to the boundary in this setting with a purely comparison-based approach forbidding other operations. Such an algorithm is presented in Section 8.5. To overcome this limitation, the number of markers is fixed and additionally a constant number of "statistics" is allowed, for example counting the number of elements smaller/larger than a marked element but also loop counters and other bookkeeping variables. In all presented algorithms the number of these additional variables is clearly in the order of the number of markers, so they are omitted in general in the analyses.

The measure for the quality of an algorithm is the minimum distance $d(x)$ to the boundary (left boundary is 1, right boundary is the number of elements) of the index of the returned element x in the sorted data. Formally it is defined as follows.

Definition 8.1. An element x in a list L is of *rank* r if and only if L contains $r-1$ elements less or equal x, and $|L|-r$ elements greater or equal x. *rank*

Note that the rank of an element is uniquely defined only if the list L contains this element exactly once.

Definition 8.2. The *boundary distance* $d(x)$ of an element x in a list L is defined as *boundary distance*

$$d(x) := \max_{k=1,\ldots,|L|} \{\min\{k-1, |L|-k\} \mid x \text{ is of rank } k\}\,.$$

Since the list will here always be the stream, the parameter L is omitted. The *boundary distance* of an algorithm A is defined as

$$d(A,n) := \min_{L,|L|=n} d\left(A(L)\right).$$

If not mentioned otherwise, the number of elements in the stream is denoted by n.

The median m has optimal distance

$$d(m) = \min\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right) = \left\lfloor \frac{n}{2} \right\rfloor,$$

representing the best value possible for d. On the opposite side of the quality scale we have the comparison-based Algorithm 2 from the next section with $d(\text{Algorithm } 2, n) = \lfloor s/2 \rfloor$ for a constant s and arbitrary large n.

8.5 A Comparison-Based Algorithm

The following algorithm simply stores the s smallest elements from the stream. One starts by storing the first s elements in sorted order. Whenever a new element appears in the stream which is smaller than one of the stored elements, it must be smaller than the maximum m_l. Therefore m_l is replaced by the new stream element. In the end, the s stored elements are sorted and their median is returned.

For a fixed s, the complete Algorithm 2 can be implemented using only comparisons and no variables other than $m_1, \ldots, m_s$.

```
1 store the first s stream elements in the variables m_1, ..., m_s
2 while end of stream not reached do
3 |   l ← argmax_{1≤i≤s} m_i
4 |   if top < m_l then
5 |   |   m_l ← top
6 |   pop
7 sort m_1, ..., m_s in ascending order
8 return m_{⌈s/2⌉}
```

Algorithm 2: This algorithm can be implemented using only comparisons for a constant s.

Observation 8.1. *Algorithm 2 returns the median of the smallest s of n elements, thus guaranteeing $\forall n \geq s : d(\text{Algorithm } 2, n) = \lfloor s/2 \rfloor$.*

9. UPPER BOUNDS

9.1 A General Upper Bound

We start with the following theorem, destroying all hope for very good approximations.

Theorem 9.1. *Every deterministic algorithm which returns an element from a stream of known size n with distance to the median at most n/α has to store at least $\alpha/4$ stream elements.*

Proof. Assume an algorithm wants to achieve a distance to the median of at most n/α. Stop after the first $\lceil n/2 \rceil$ elements were read from the stream. We call the set containing these elements M. None of the elements in M can be ruled out being the median—just let the values of the following $\lfloor n/2 \rfloor$ elements be smaller/larger properly. Covering the sorted order of M completely with intervals of the desired distance requires storing at least every $2n/\alpha$th element in the sorted order. This results in a total of

$$\left\lceil \frac{n}{2} \right\rceil \frac{\alpha}{2n} \geq \frac{\alpha n}{4n} = \frac{\alpha}{4}$$

stored elements. □

This immediately implies the following result for constant storage.

Corollary 9.2. *No algorithm which stores only a constant number of elements from a stream can achieve an approximation of the median's position in the sorted data which is sub-linear in the total number of elements n. Formally this shows for all algorithms A with constant storage: $d(A, n) = n/2 - \Omega(n)$.*

9.2 Obtaining Bounds by Playing Games with One Marker

After the general upper bound from the previous section, we will provide some explicit upper bounds for very small storage, namely one or two markers. We prove an upper bound on d over all possible algorithms by modeling the situation as an *adversary game*: The adversary is the stream which selects the next element. The algorithm has to decide whether this element becomes ignored or marked, releasing one of the formerly marked elements.

adversary game

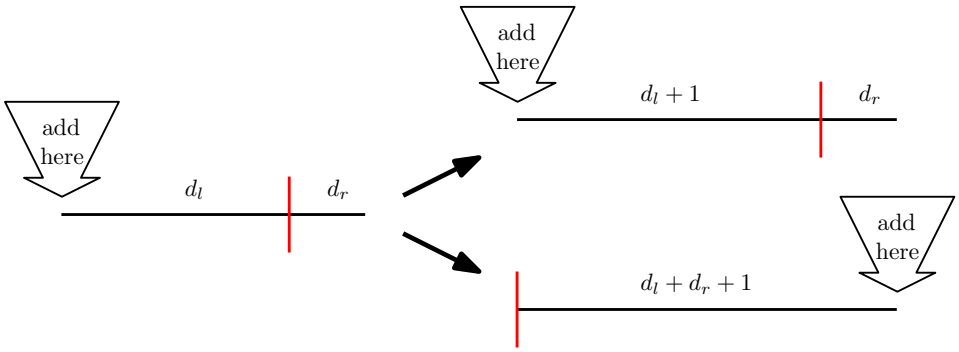


Fig. 9.1: The horizontal line represents the sorted order of the stream elements seen so far, the vertical line is the marker.

This game continues indefinitely to simulate the asymptotic result on n. This might be considered as the online version of the median problem.

For every algorithm that uses only a single marker, the adversary can choose a data stream in such a way that the marker is always the smallest or largest element. This is true, even if the algorithm is allowed to "know" all elements seen so far and can consider them in its computation, as long as it is only allowed to set the marker to new elements. Figure 9.1 shows the idea behind such an adversarial stream: The adversary only adds new extrema. The algorithm might decide to keep its marker which remains close to the boundary (upper right). If the marker is moved to the new element, it will be stuck at the boundary by having the adversary switch the insertion position to the other end (lower right). Since the distances d_l and d_r from Figure 9.1 are initially 0, the following observation holds.

Observation 9.3. *For an adversarial data stream a single marker will always end at the boundary.*

9.3 Obtaining Bounds by Playing Games with Two Markers

adversary game

For algorithms using two markers we will also show an upper bound by an *adversary game*. In this case the knowledge about all elements seen so far, as used for algorithms with one marker, is too mighty and would lead to a variant we will discuss in Section 9.4. This strength follows from the knowledge of the exact position of a new element in the sorted elements seen so far. This allows to determine exactly how a new element partitions the elements between two markers.

We model the game for algorithms using two markers in such a way, that the adversary only specifies the interval between two markers or a marker and the boundary where the next element will lie in. The algorithm ignores the new element or selects a marker which is set to that position. Afterwards

the adversary reveals the exact position. A bound for this scheme with two markers m_1, m_2 is depicted in Figure 9.2 and explained in the proof for the following lemma.

Lemma 9.4. *For two markers* m_1, m_2, $\max(d(m_1), d(m_2)) \in O(\sqrt{n})$ *holds in the worst case.*

Proof. Assume $d(m_1) = d(m_2) = d$ (const) in the beginning. The adversary adds an element in the interval containing the median (top left in Figure 9.2)). Not moving the marker neither increases $d(m_1)$ nor $d(m_2)$ (top right), leading to an arbitrarily bad result. Therefore the algorithm has to move a marker eventually.

After one marker was set, the adversary reveals that it is next to the other marker (middle left). We have $d(m_1) = d + 1$ or $d(m_2) = d + 1$. This is the only increase of $d(m_1)$ or $d(m_2)$ in the scheme. Now adding new extrema either leaves $d(m_1)$ and $d(m_2)$ unchanged (middle right) or shifts a marker to the boundary (bottom left).

Having, without loss of generality, $d(m_1) = d + 1$ and $d(m_2) = 0$ as in the bottom left case in Figure 9.2 allows the insertion of $d+1$ elements next to m_2, ending in the situation depicted in the bottom right case.

In total, the boundary distance has increased by one, while $d+3$ elements were inserted. Now we can apply the same scheme again with $d + 1$ instead of d. To consume all n elements in r rounds, starting with $d = 0$ we have

$$\sum_{d=0}^{r-1} (d + 3) = 3r + \frac{r(r-1)}{2} = \frac{r^2 + 5r}{2} \approx n.$$

It clearly follows that $r \in \Theta(\sqrt{n})$ and we have $d(m_1), d(m_2) \leq r$ and therefore $d(m_1), d(m_2) \in O(\sqrt{n})$. Note that the case distinction is not complete because some obviously bad choices are omitted. □

The idea of revealing the true position later already fails for three markers. For an arbitrary fixed number of markers it is not known whether the linear bound can be reached or not. The described adversary game seems to be a bad choice for a proof for bigger storage, because the rules of the game would have to become much more involved to obtain sensible results.

9.4 A Variant of the Online Problem

For offline problems the complete input is available during the whole algorithm. *Online problems*, by contrast, reveal only small pieces of the input one after another and the algorithm has to make decisions based on incomplete data. This is a much stronger model than our streaming model with constant memory because all elements seen once are still available. Therefore every solution for the streaming problem is also a solution for the online

online problem

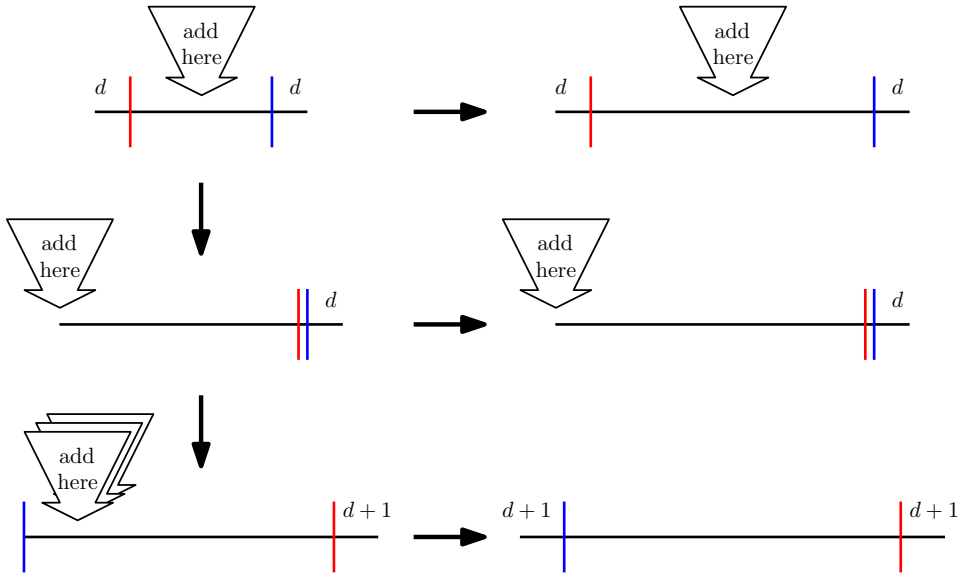


Fig. 9.2: Beating a two marker scheme.

problem and every upper bound for the online problem is an upper bound for the streaming problem.

An obvious and simple algorithm is to compute the exact median over the elements seen so far after each received element until the input stream ends. A more interesting and challenging problem formulation is the following.

The input is a stream of unknown size n, the streamed objects are pairs (t_i, r_i) where t_i is an element as before and r_i is a rank of t_i among the elements $t_1, t_2, \ldots, t_i$.

secretary problem

This formulation includes a part of the additional knowledge for the online problem in the streaming model and is very close to the formulation of the *secretary problem* [CMRS64], see Section 8.3.1. This additional knowledge is already sufficient to get a linear approximation of the median with two markers with the following algorithm.

For simplicity we assume that the stream contains at least two elements.

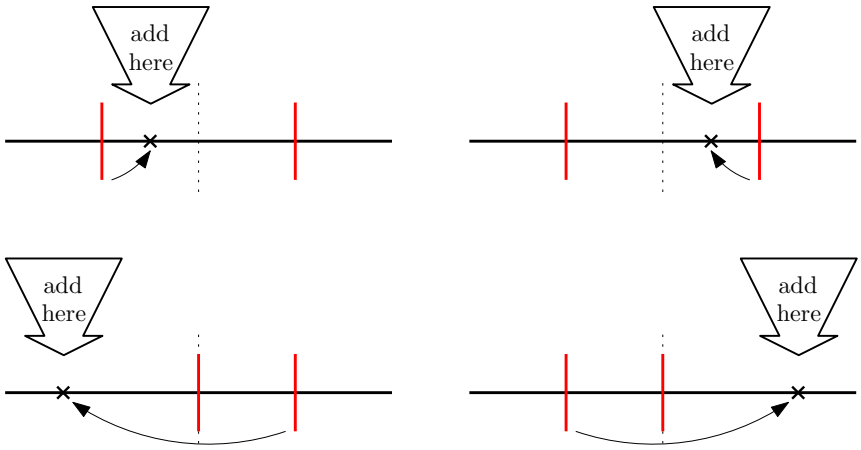


Fig. 9.3: The four interesting cases for the online algorithm with two markers. The solid vertical lines are the markers, the dotted line indicates the median. The bottom two cases only occur, if one marker points exactly at the median. If no case applies, the markers are not moved.

```
1  m_1 ← pop.t
2  m_2 ← pop.t
3  if m_1 > m_2 then swap m_1 and m_2
4  r_1 ← 1
5  r_2 ← 2                    // loop invariants: r_1 < r_2 and m_1 ≤ m_2
6  for i = 2, 3, 4, ... do
7      if top.t ≤ m_2 then r_2 ← r_2 + 1 ;     // maintain rank of m_2
8      if top.t ≤ m_1 then
9          r_1 ← r_1 + 1                       // maintain rank of m_1
10         if r_1 ≥ ⌈i/2⌉ then                    // bottom left case
11             m_2 ← top.t
12             r_2 ← top.r
13     else if r_1 < top.r ≤ ⌈i/2⌉ then            // top left case
14         m_1 ← top.t
15         r_1 ← top.r
16     else if ⌈i/2⌉ < top.r < r_2 then            // top right case
17         m_2 ← top.t
18         r_2 ← top.r
19     else if r_2 < top.r and r_2 < ⌈i/2⌉ then   // bottom right case
20         m_1 ← top.t
21         r_1 ← top.r
22     if m_1 > m_2 then swap m_1 and m_2 and swap r_1 and r_2
23     if pop is end of stream then
24         if r_1 ≥ i − r_2 then return m_1 else return m_2
```

Algorithm 3: Achieving a linear approximation of the median with only two markers m_1, m_2 by knowing their rank.

Algorithm 3 handles the four important cases as depicted in Figure 9.3. The horizontal lines represent the elements from the stream seen so far in sorted order. The solid vertical bars are the markers and the dotted vertical bar in the middle shows the position of the median. The big arrow indicates the rank of the next stream element.

The main idea of the algorithm is to keep the median between the two markers while minimizing their distance. The distance between the markers is reduced if possible, as seen in the two top cases of Figure 9.3. If there is a chance that both markers are on the same side of the median, the marker further away is moved to the other side, shown in the two bottom cases of Figure 9.3. In the end, the marker closer to the exact median is returned.

Theorem 9.5. *Algorithm 3 returns an element with a rank between $n/4$ and $3n/4$ from a stream with rank information using two markers.*

Proof. For simplicity of notation, we assume that all elements and thereby their ranks, are unique. Otherwise one can fix any possible rank for elements with non-unique rank. We will prove that for i elements holds:

$$\text{rank}\,(\min(m_1, m_2)) \leq \frac{i}{2} \leq \text{rank}\,(\max(m_1, m_2)) \tag{9.1}$$

$$|\text{rank}\,(m_2) - \text{rank}\,(m_1)\,| \leq \frac{i}{2} \tag{9.2}$$

The first claim (9.1) is very easy to show because the only way a marker can switch its side relative to the median, say from left to right, is by adding elements on the left side. Exactly this is handled in the two bottom cases in Figure 9.3 by moving the second marker to the opposite side. These cases only occur if one marker is the exact median.

Due to the invariant we have $m_1 < m_2$, so for (9.2) it remains to show that $\text{rank}\,(m_2) - \text{rank}\,(m_1) \leq i/2$. Starting the induction with $i = 2$ elements, the distance between the markers is exactly 1 and the relation is true. Obviously the two top cases in Figure 9.3 only decrease the distance between the markers, while increasing i. Adding an element left of m_1 or right of m_2 does not influence the distance but increases i. In the two bottom cases one marker is the median and the other marker is set at most to the boundary, fixing the distance between the markers to exactly $i/2$ and therefore fulfilling (9.2).

Since rank information is provided for every element, it is easy to maintain the rank of the marked elements and to return the one with larger boundary distance. If one marker has boundary distance smaller than $n/4$, e.g. because its rank is smaller than $n/4$, $|\text{rank}\,(m_2) - \text{rank}\,(m_1)\,| \leq n/2$ holds and therefore the other boundary distance must be larger than $n/4$. □

10. LOWER BOUNDS AND ALGORITHMS

10.1 Dealing with Unknown Data Size

Every algorithm for streams with unknown total size is obviously also an algorithm for streams with known size n. The following algorithm achieves a distance of $\Omega(\sqrt{n})$ to the boundary without using n. It uses two markers and is therefore asymptotically optimal due to Lemma 9.4. It is the best we can achieve without using n and this algorithm acts as a building block for the improved algorithms in the next sections.

The general idea is to have roughly $\sqrt{n}$ rounds. Each round increases the distance to the boundary by at least one.

As mentioned in Section 8.4, two markers alone do not suffice for any interesting result. A constant number of additional variables are required for counting and general bookkeeping. These are omitted in the analysis.

```
   UnknownSize-Sqrt:
1  m1 ← pop
2  for r = 2, 3, 4, ... do
3  |  m2 ← top
4  |  repeat r times
5  |  |  pop
6  |  |  if end of stream reached then return m1
7  |  |  if (m2 < top < m1) or (m2 > top > m1) then m2 ← top
8  |  if last r elements were all smaller or all larger than m1 then
9  |  |  m1 ← m2
```

Algorithm 4: Achieving a distance of at least $\sqrt{2(n+1)} - 3$ to the boundary with only two markers m_1, m_2.

Theorem 10.1. *Algorithm 4 returns an element with distance to the boundary in the sorted data at least $\sqrt{2(n+1)} - 3$ without knowing n beforehand, using two markers.*

Proof. Every cycle of the **for** loop is considered as a round. We will show that the boundary distance after round r is at least $r - 1$. The proof is by induction.

For $r = 1$ the list of seen elements contains only the first element m_1 which trivially has a boundary distance of 0 in this list.

For rounds $r > 1$ the marker has boundary distance at least $r - 2$ in the beginning and several cases are possible.

1. The considered r elements are all larger than m_1. The marker m_1 is set to the minimum m_2 of these r elements. This guarantees that at least $r - 1$ elements are still larger than m_2, which corresponds to the new distance to the right boundary. On the other hand, the number of elements smaller than m_2 is at least the number of elements smaller than m_1, since $m_2 > m_1$, plus the additional element m_1. Therefore the new distance to the left boundary is larger than $r - 2$ by at least one.

2. The considered r elements were all smaller than m_1. This case is symmetric to the first case.

3. Otherwise. At least one element smaller than m_1 and one element larger than m_1 is added. This increases the boundary distance of m_1 on both sides by at least one.

In all three cases the boundary distance on both sides is either increased by at least one or set directly to at least $r - 1$. Thus the boundary distance is at least $r - 1$ after round r.

If the input stream terminates in the middle of a round, the elements from this round are ignored and the marker m_1 is not updated. Hence we have to count the number of completed rounds k for $n \geq 2$ elements. Since the $k + 1$st round was not completed, the number of elements consumed in the first k rounds is at least $n - k$. We have

$$n - k \leq \sum_{r=1}^{k} r \Rightarrow n \leq \frac{k^2 + 3k}{2} \Rightarrow k \geq \sqrt{2n + \frac{9}{4}} - \frac{3}{2} > \sqrt{2(n+1)} - 2. \qquad \square$$

This result directly implies the following sharp bound.

Corollary 10.2. *The achievable boundary distance with two markers is* $\Theta(\sqrt{n})$.

Proof. This follows immediately from Theorem 10.1 and Lemma 9.4 and is realized by algorithm UnknownSize-Sqrt. $\square$

In practice, a new round should already start when at least one element smaller than the median and one element larger than the median was read. In the worst case in every round all but the last element are on the same side of the median. Therefore this optimization has no impact on the worst-case running time and is ignored in the algorithm to simplify the analysis.

10.2 Improvements for Known Data Size

If the number of input elements is not known, an algorithm has to maintain a good approximation all the time. Hence it gives more algorithmic freedom to know the total number of elements n in advance. We use this freedom to process specified fractions of the input as a block. Four markers are used.

We split the input into at most n/t blocks of size roughly t. Two consecutive blocks are processed together in one round. From the first block we compute an approximation of its median using algorithm UnknownSize-Sqrt from Section 10.1. The next block is used to verify the quality of the approximation, which is then refined in the next round. For this purpose a lower threshold *low* and an upper threshold *high* are maintained, such that there are always enough elements above *low* and below *high*. These two thresholds are usual markers. They are used to filter the input for UnknownSize-Sqrt: Only elements in the range between *low* and *high* are given as input to UnknownSize-Sqrt.

To make use of our filters, we replace the function `pop` in UnknownSize-Sqrt by the extended version `pop*`.

```
   pop*:
1  loop indefinitely
2  |  if end of the stream is reached then
3  |  |  if l ≥ h then output low else output high
4  |  |  stop
5  |  if top < low then l ← l + 1; pop
6  |  else if top > high then h ← h + 1; pop
7  |  else return pop
```

Let $K = \Theta(\sqrt{t})$ be a constant representing the guaranteed boundary distance from algorithm UnknownSize-Sqrt according to Theorem 10.1. This constant is added to the number of elements smaller than *low* or larger than *high* if the respective filter is changed after a call to UnknownSize-Sqrt because the result of UnknownSize-Sqrt has a guaranteed distance to the filter values of K by Theorem 10.1.

```
  KnownSize(n,t):
1 low ← −∞                    // invariant:  l ≤ rank(low) ≤ n − t/2
2 high ← ∞                    // invariant:  n − h ≥ rank(high) ≥ t/2
3 l ← 0
4 h ← 0
5 loop indefinitely
6 |  m_1 ← use UnknownSize-Sqrt on t elements with pop*
7 |  c ← 0
8 |  for the next t elements do
9 |  |  if pop* ≥ m_1 then c ← c + 1
10|  if c > t/2 then low ← m_1; l ← l + K + t − c
11|  else high ← m_1; h ← h + K + c
```

Algorithm 5: Achieving a boundary distance of $\Omega\left(n^{2/3}\right)$ for $t \in \Theta\left(n^{2/3}\right)$ with four markers.

Lemma 10.3. *While processing any $3t$ consecutive elements from the stream, the number of elements known to be smaller than low plus the number of elements known to be larger than high, which are counted in l and h respectively, increases by $\Omega\left(\sqrt{t}\right)$.*

Proof. Two cases are possible for the $3t$ elements.

1. If line 10 is not reached, more than t elements are smaller than *low* or larger than *high* and are hence filtered out. Therefore $l + h$ increased by at least t in the subroutine `pop*`, either in the for-loop or in UnknownSize-Sqrt.

2. Otherwise a set B of at least t elements pass the filter. Algorithm UnknownSize-Sqrt returns an element m_1 with distance $\Omega\left(\sqrt{t}\right)$ to both boundaries of B by Theorem 10.1. Counting over the next t elements allows two symmetric cases, so we consider only the case that at least $t/2$ elements are not smaller than m_1. The algorithm sets *low* to m_1 in line 10, guaranteeing the claimed increase of $\Omega\left(\sqrt{t}\right)$ in l. □

Lemma 10.4. *Algorithm* KnownSize *returns an element with boundary distance $\Omega\left(\min\left\{t, \sqrt{t}n/t\right\}\right)$ if n is known beforehand. Four markers are used.*

Proof. The algorithm will only change *low* if at least $t/2$ larger elements are known, so the distance between *low* and the maximum is always at least $t/2$. We only have to care about pushing *low* as far away from the minimum as we can to obtain a large distance to both boundaries. The symmetric statement holds for *high*.

The n input elements can be seen as $n/(3t)$ blocks with $3t$ elements each. Lemma 10.3 gives a total distance of

$$l + h \geq \frac{n}{3t}\Omega\left(\sqrt{t}\right) = \Omega\left(\sqrt{t}\frac{n}{t}\right)$$

to both boundaries in the sorted data. At least half of this distance must be attained by *low* or *high* which is returned respectively.

The marker m_1 in this algorithm is the same as m_1 in the subroutine, so two markers are used by the subroutine plus two for *low* and *high*. □

For the optimal result we want to maximize the minimum in the expression $\Omega\left(\min\left\{t, \sqrt{t}n/t\right\}\right)$ by setting up the following equation.

$$t = \sqrt{t}\frac{n}{t} = \frac{n}{\sqrt{t}} \Rightarrow t^{3/2} = n \Rightarrow t = n^{2/3}$$

Corollary 10.5. *Algorithm* KnownSize *returns an element with boundary distance* $\Omega\left(n^{2/3}\right)$ *if* n *is known beforehand. Four markers are used.*

Proof. Apply Lemma 10.4 with $t = n^{2/3}$. □

Theorem 10.6. *A distance to the boundary of* $\Omega\left(c_a n^{1-\frac{1}{a+1}}\right)$ *can be achieved with* $2a$ *markers for known* n*. The constant factor* c_a *solely depends on* a *and not on* n*.*

Proof. Instead of using algorithm UnknownSize-Sqrt as a subroutine, one can use algorithm KnownSize recursively. This needs two additional markers for the *low* and *high* values in each recursive step. Assume, we have a subroutine which guarantees a boundary distance of $\Omega(n^x)$. Computing the optimal value for t by a similar formula as in the proof of Lemma 10.4 resolves to

$$t = t^x\frac{n}{t} \Rightarrow t^{2-x} = n \Rightarrow t = n^{\frac{1}{2-x}}.$$

Using this subroutine and this value of t in the proof of Lemma 10.4 yields an algorithm with boundary distance $\Omega\left(n^{1/(2-x)}\right)$. In particular for a subroutine with guarantee $\Omega\left(n^{a/(a+1)}\right)$ the guarantee is raised to

$$\Omega\left(n^{\frac{1}{2-\frac{a}{a+1}}}\right) = \Omega\left(n^{\frac{a+1}{a+2}}\right).$$

Starting with $x = 1/2$ from algorithm UnknownSize-Sqrt, $a/(a+1)$ is reached after $a-1$ recursive levels, each requiring two markers, $2a$ in total.

The results for the algorithms applied in each recursive level are asymptotic ones. Hence they contain "hidden" constants which are accumulated in the factor c_a. □

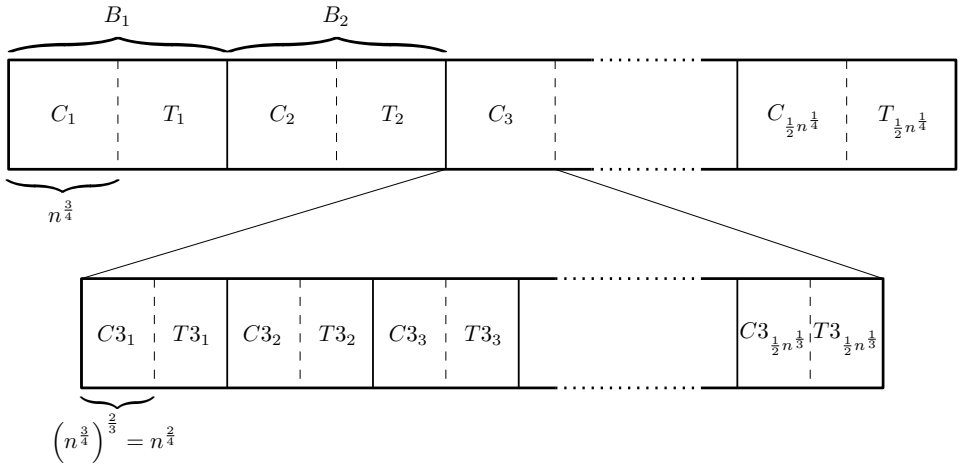


Fig. 10.1: The division of the input of size n into computation and testing blocks (C_i and T_i) by algorithm KnownSize is illustrated by this example with guaranteed boundary distance $\Omega\left(n^{3/4}\right)$.

The idea of the nested application of algorithm KnownSize which leads to the results from Theorem 10.6 is illustrated in Figure 10.1. It shows how the original input is divided into double-blocks B_i. In each block, the first part C_i is used to compute a new candidate, while the second part T_i tests how well the candidate splits this block. The new element in the recursive idea is that each block C_i itself is again split into blocks B_{i_j} which are processed similar to the B_i. The size t of these blocks has to be computed properly, as seen in the proof of Theorem 10.6.

10.3 Transferring the Results for Known Data Size to Unknown Data Size

unbounded search

The algorithms from the previous section require that the total number of elements is known beforehand. To make them applicable for stream with unknown size, a standard technique, known as *unbounded search* [BY76], can be used. Although the original paper by Bentley and Yao [BY76] and also later proofs [Bei90] achieve a more involved bound, the simple algorithm below suffices for our needs.

```
  UnknownSize-Eps:
1 k ← 1
2 repeat
3 |   m ← KnownSize(k,k^ε)
4 |   k ← 2k
5 until stream ends
6 return m
```

Algorithm 6: Applying an algorithm for known stream size repeatedly on chunks of increasing size.

In each round, the algorithm for known size is applied to the sequence of the next k elements from the stream for a fixed k. Starting with a constant, here one, the value for k is doubled each time the algorithm provides a result.

Theorem 10.7. *Given an algorithm A for streams with known size n and guaranteed boundary distance $d(A,n)$. If the function $d(A,n)$ has a lower bound $f(n)$ such that $f(n)/n$ is monotone decreasing then it is possible to create an algorithm B for streams with unknown size with $d(B,n) \geq f(n)/4$ and with the same number of markers plus one.*

Proof. The proof is by induction. Let n be the unknown number of elements in the stream. For $n = 1$ the exact median of the single element is returned by algorithm UnknownSize-Eps and $d(A,1) = f(1) = d(B,1) = 0$. This is considered as the initialization, round zero. For round $r \geq 1$ the number of elements to be processed in this round is $k = 2^r$. The total number of elements consumed from the stream so far is the geometric series

$$\sum_{j=0}^{r-1} 2^j = 2^r - 1.$$

If $2^r - 1 + k = 2^{r+1} - 1 \leq n$ holds, the algorithm will continue to the next round. In case the stream ends in the middle of round r, $n < 2^{r+1}$ must hold. The returned value m was obtained in round $r-1$ and therefore we have

$$d(B,n) = d(m) \geq d\left(A, 2^{r-1}\right) = d\left(A, 2^{r+1}/4\right) \geq d(A, n/4).$$

Since $f(n)$ is a lower bound for $d(A,n)$, we have

$$d(B,n) \geq d(A,n/4) \geq f(n/4) \geq f(n)/4$$

because $f(n)/n$ is monotone decreasing and thus giving

$$\frac{f(n/4)}{n/4} \geq \frac{f(n)}{n} \Rightarrow 4f(n/4) \geq f(n).$$

□

A direct implication of Theorem 10.7 is the following.

Corollary 10.8. *A distance to the boundary of* $\Omega\left(c_a n^{1-\frac{1}{a+1}}\right)$ *can be achieved with* $2a+1$ *markers for unknown* n*. The factor* c_a *solely depends on* a *and not on* n*.*

Proof. Algorithm KNOWNSIZE guarantees a boundary distance of $\Omega\left(c_a n^{1-\frac{1}{a+1}}\right)$ for $2a$ markers by Theorem 10.6. A lower bound for $d(\text{KNOWNSIZE}, n)$ is therefore $f(n) = kc_a n^{1-\frac{1}{a+1}}$ for a suitable constant k and n large enough. The function $f(n)/n = kc_a n^{-\frac{1}{a+1}}$ is obviously monotone decreasing. The corollary follows from Theorem 10.7. □

11. EXTENSIONS

11.1 Arbitrary Quantiles

A more general problem than finding the median is to find an element with a fixed rank k. Although it contains the median problem as a special case for $k = \lceil n/2 \rceil$, the input seems to be fairly different. Without knowledge about n, $k = \lceil n/2 \rceil$ is just a number—the algorithm does not know that it is the median which it has to approximate. A specialized median approximation algorithm on the other hand is just optimized to maximize the boundary distance.

Nevertheless, the arbitrary quantile problem cannot be significantly simpler than the median problem, as is easy to observe.

Lemma 11.1. *Given an algorithm $A(k)$ which computes an α-approximation for the element of rank k in a stream of n elements. This means $A(k)$ returns an element x with $\|rank(x) - k\| \leq \alpha n$. From A, one can derive a 4α-approximation algorithm for the median problem.*

Proof. We apply the doubling strategy from Theorem 10.7. In round r the number of considered elements is 2^r. $A(k)$ is called with $k = 1/2 \cdot 2^r = 2^{r-1}$. The 4α-approximation for the median problem is now obtained by the very same argument about the total number of elements in the stream as in the proof of Theorem 10.7. □

Unfortunately the problem formulation for rank k with k as a numeric parameter is not useful in our case, because our approximations are asymptotic results. Therefore we cannot provide a factor α which bounds the distance of the rank of the returned element to the one requested.

11.2 Multiple Passes

The algorithms from the prior sections give approximations of the median in a single pass. It is fairly easy to apply these simple algorithms repeatedly to shrink the range of candidates for the value in question, i.e. the median or the element of *rank* k, in each iteration and finally obtain the exact value after several passes. In this section the number of required passes is analyzed. Problems of that kind typically require $\Theta(\log n)$ "passes" for random-access models.

rank

The approximation m obtained after a single pass splits the original set into two sets, one with values smaller than m and the others larger. Only one of these two sets can contain the element in question, so dropping the other one reduces the number of elements. This is repeated until the remaining set completely fits into the constant storage and the desired element is deduced directly. Since it is not known whether the requested element is smaller or larger than the approximation m, every second pass is used to count the number of smaller/larger elements relative to m which reveals the rank of m and thereby determines which set to drop. By remembering the number of discarded elements, the rank of the requested element in the reduced set can be computed.

The analysis treats the case where n is known beforehand because n is known after the first pass anyway.

rank

Theorem 11.2. *For any fixed storage size $2s+2$, the element of* rank k *in a set will be found after $O\big(n^{1/s}\big)$ passes.*

Proof. The idea is to apply algorithm KnownSize repeatedly. Let the function $P(n)$ describe the number of passes necessary for n elements. Each pass removes at least $c_1 n^{1-1/s}$ elements for some fixed $c_1 > 0$ by Theorem 10.6, so we have $P(n) \le P(n - c_1 n^{1-1/s}) + 2$. We will show $P(n) \le c_2 n^{1/s}$ for a fixed $c_2 > 0$ by induction.

$$P(n) \le P\left(n - c_1 n^{1-\frac{1}{s}}\right) + 2 \le c_2 \left(n - c_1 n^{1-\frac{1}{s}}\right)^{\frac{1}{s}} + 2$$

$$= c_2 n^{\frac{1}{s}} \left(1 - \frac{c_1}{n^{\frac{1}{s}}}\right)^{\frac{1}{s}} + 2 \overset{!}{\le} c_2 n^{\frac{1}{s}}$$

$$\Leftrightarrow \left(1 - \frac{c_1}{n^{\frac{1}{s}}}\right)^{\frac{1}{s}} + \frac{2}{c_2 n^{\frac{1}{s}}} \overset{!}{\le} 1 \Leftrightarrow 1 - \frac{c_1}{n^{\frac{1}{s}}} \overset{!}{\le} \left(1 - \frac{2}{c_2 n^{\frac{1}{s}}}\right)^{s}$$

For $c_2 = 2s/c_1$ this follows from Bernoulli's inequality

$$1 - sx \le (1-x)^s \text{ with } x = \frac{c_1}{s n^{\frac{1}{s}}} < 1, s \ge 1$$

and since s and c_1 are fixed, c_2 is constant.

The number of markers needed is the number of markers needed for the algorithm KnownSize plus two additional markers to simulate the removal of the $c_1 n^{1-1/s}$ elements. This can easily be done using a lower and an upper filter as in algorithm KnownSize. □

Theorem 11.2 bases on the simple scheme described above. Only a single marker from one pass influences the next pass. Possibly, better results can be obtained by using more results from previous passes in a cleverer way.

12. EXPERIMENTAL RESULTS

As a proof of concept, the proposed algorithms have been implemented in Java. The algorithms are implemented as given in this thesis and are not tweaked any further. In the respective sections one can find hints, how to optimize the algorithms for practical purposes.

The most interesting quantity for a median approximation algorithm is how far the returned element is away from the median. This is measured as the difference between the ranks. For better comparability it is given as relative value in percent with respect to the total number of elements. This value is independent of the distribution which only influences the concrete values but not the ranks of the stream elements.

Table 12.1 shows the most important results at a glance. Each line shows the average results over 50 independent random streams with the specified number of elements for three algorithms. The first "algorithm" called "random" just picks a random stream element while UnknownSize-Sqrt and KnownSize are the respective algorithms from this thesis. The values in this table show the relative distance to the median in percent. For example the element of rank 30 among 101 elements has a relative distance to the median of 20% and splits the elements in a ratio of 29:71.

The results for the randomly picked element are very close to the expected value of 25%, independent of the number of elements. There are two important observations to be made from Table 12.1: Both algorithms perform very well and the quality seems to be independent of or even increasing with number of elements. Algorithm UnknownSize-Sqrt allows distances to the median of $\lfloor n/2 \rfloor - \sqrt{2(n+1)}$ by Theorem 10.1. This would give worst-case relative distances of more then 45% for 1000 elements and even 49%

number of elements	random	UnknownSize-Sqrt	KnownSize
1000	23.4	10.6	5.8
2000	26.4	10.3	4.6
10000	25.3	13.6	2.0
100000	26.1	10.3	1.1
1000000	23.8	11.1	1.0

Tab. 12.1: Average relative distances to the median in % over 50 runs with the specified numbers of elements.

for 1000000 elements, increasing with the number of elements. Nevertheless UnknownSize-Sqrt provides an element with a relative distance of slightly more then 10% on average in practice. Algorithm KnownSize even seems to become better with an increasing number of elements, down to 1% for 1000000 elements. This is again tremendously better than the worst-case and a substantial improvement over UnknownSize-Sqrt.

The following three tables show more detailed results for individual experiments. They show 35 runs with the respective number of stream elements drawn uniformly at random.

The columns in the table have the following meaning. The first column shows the *number of the experiment*. This determines the stream elements completely, so results from different algorithms become comparable. The *number of elements* is self-explanatory. The *split ratio in %* shows the percentage of elements smaller than the element returned by the algorithm, then a colon, and then the percentage of elements larger than the returned element. This always adds up to 100. The *split ratio average* in the last row is slightly different. It shows the ratio between the average of the smaller value from each row in the "split ratio" column to the average of the larger values. Otherwise the symmetric errors on both sides would cancel each other, rendering the average value useless.

The *distance to the median* is the distance between the rank of the returned element and $\lceil n/2 \rceil$ (absolute) and this distance divided by n in percent (relative). The total absolute distance lies between 0 and $\lfloor n/2 \rfloor$ while the relative distance is between 0 and 50. The number of *candidate changes* shows how often the algorithm changes the element it would return if the stream terminates. This column is not available for the random element because this is obviously picked exactly once.

The quantities in the tables are compared for a randomly picked element, the result of UnknownSize-Sqrt and the result of KnownSize. As expected, the randomly picked element can be arbitrarily bad, as in line 2 and line 11 of Table 12.2, but can also be very good, as in line 17 and 30 of Table 12.2.

Algorithm UnknownSize-Sqrt provides much better results. The worst-case relative distance to the median of 33.5% in line 4 (Table 12.3) is much better than the worst-case and still far away from the average of 11.7%. Since they can be achieved algorithmically with nearly no resources, it is definitely a valuable improvement over the 25% average relative distance of a random element.

As a surprise, the KnownSize algorithm lead to a substantial improvement over UnknownSize-Sqrt. With its worst-case relative distances to the median of 15.9% (Table 12.4) and an average of 2.7% it is far away from the possible worst-case values. Relative distances of more than 5% only occur for experiments with few elements. If these results are not sufficient for applications, there is still room for improvements since the algorithm only uses one recursive step and four markers, see Section 10.2.

no.	number of elements	split ratio in %	distance to median absolute	distance to median relative
1	1000	70.3 : 29.7	203	20.3
2	1000	0.9 : 99.1	491	49.1
3	1000	29.2 : 70.8	208	20.8
4	1000	59.3 : 40.7	93	9.3
5	1000	62.0 : 38.0	120	12.0
6	1000	74.3 : 25.7	243	24.3
7	1000	1.4 : 98.6	486	48.6
8	1000	63.8 : 36.2	138	13.8
9	1000	76.2 : 23.8	262	26.2
10	1000	23.8 : 76.2	262	26.2
11	10000	1.3 : 98.7	4868	48.7
12	10000	70.9 : 29.1	2086	20.9
13	10000	16.9 : 83.1	3307	33.1
14	10000	61.1 : 38.9	1110	11.1
15	10000	98.4 : 1.6	4840	48.4
16	10000	36.3 : 63.7	1373	13.7
17	10000	52.1 : 47.9	207	2.1
18	10000	70.9 : 29.1	2092	20.9
19	10000	2.7 : 97.3	4730	47.3
20	10000	15.9 : 84.1	3409	34.1
21	100000	38.5 : 61.5	11540	11.5
22	100000	86.8 : 13.2	36817	36.8
23	100000	60.2 : 39.8	10192	10.2
24	100000	85.0 : 15.0	35043	35.0
25	100000	29.4 : 70.6	20558	20.6
26	100000	29.9 : 70.1	20111	20.1
27	100000	32.6 : 67.4	17417	17.4
28	100000	73.9 : 26.1	23854	23.9
29	100000	77.4 : 22.6	27440	27.4
30	100000	51.2 : 48.8	1225	1.2
31	1000000	79.7 : 20.3	296710	29.7
32	1000000	62.4 : 37.6	123758	12.4
33	1000000	43.8 : 56.2	61935	6.2
34	1000000	12.6 : 87.4	373637	37.4
35	1000000	4.1 : 95.9	458963	45.9
average		25.2 : 74.8		24.8

Tab. 12.2: Experimental results for picking a random element from the stream.

no.	number of elements	split ratio in %	distance to median absolute	distance to median relative	candidate changes
1	1000	71.3 : 28.7	213	21.3	2
2	1000	40.3 : 59.7	97	9.7	2
3	1000	58.8 : 41.2	88	8.8	2
4	1000	16.5 : 83.5	335	33.5	3
5	1000	62.6 : 37.4	126	12.6	3
6	1000	63.5 : 36.5	135	13.5	1
7	1000	50.5 : 49.5	5	0.5	1
8	1000	39.0 : 61.0	110	11.0	2
9	1000	44.6 : 55.4	54	5.4	3
10	1000	54.8 : 45.2	48	4.8	2
11	10000	63.0 : 37.0	1299	13.0	1
12	10000	42.1 : 57.9	791	7.9	2
13	10000	61.2 : 38.8	1116	11.2	1
14	10000	59.9 : 40.2	985	9.9	2
15	10000	66.2 : 33.8	1620	16.2	1
16	10000	70.9 : 29.1	2086	20.9	0
17	10000	40.9 : 59.1	913	9.1	1
18	10000	34.4 : 65.6	1557	15.6	1
19	10000	50.6 : 49.4	61	0.6	2
20	10000	45.5 : 54.5	448	4.5	1
21	100000	61.1 : 38.9	11072	11.1	2
22	100000	43.7 : 56.3	6251	6.3	2
23	100000	67.1 : 32.9	17140	17.1	2
24	100000	60.8 : 39.2	10791	10.8	1
25	100000	58.1 : 41.9	8061	8.1	1
26	100000	42.9 : 57.1	7142	7.1	2
27	100000	61.3 : 38.7	11259	11.3	1
28	100000	41.8 : 58.2	8230	8.2	3
29	100000	37.9 : 62.1	12070	12.1	1
30	100000	75.2 : 24.8	25198	25.2	1
31	1000000	41.9 : 58.1	80860	8.1	2
32	1000000	40.9 : 59.1	91298	9.1	1
33	1000000	38.7 : 61.3	112749	11.3	1
34	1000000	24.9 : 75.1	250703	25.1	2
35	1000000	42.5 : 57.5	75338	7.5	1
average		38.3 : 61.7		11.7	2

Tab. 12.3: Experimental Results for algorithm UnknownSize-Sqrt.

no.	number of elements	split ratio in %	distance to median absolute	distance to median relative	candidate changes
1	1000	52.3 : 47.7	23	2.3	36
2	1000	49.4 : 50.6	6	0.6	38
3	1000	58.2 : 41.8	82	8.2	9
4	1000	34.1 : 65.9	159	15.9	37
5	1000	50.2 : 49.8	2	0.2	37
6	1000	49.8 : 50.2	2	0.2	37
7	1000	50.5 : 49.5	5	0.5	33
8	1000	39.0 : 61.0	110	11.0	36
9	1000	38.0 : 62.0	120	12.0	36
10	1000	54.8 : 45.2	48	4.8	44
11	10000	53.7 : 46.3	373	3.7	140
12	10000	45.3 : 54.7	472	4.7	3
13	10000	49.7 : 50.4	35	0.4	68
14	10000	50.8 : 49.3	75	0.8	39
15	10000	52.4 : 47.6	240	2.4	123
16	10000	51.6 : 48.4	161	1.6	35
17	10000	47.3 : 52.7	274	2.7	53
18	10000	53.0 : 47.0	301	3.0	111
19	10000	53.2 : 46.8	322	3.2	56
20	10000	49.2 : 50.8	79	0.8	128
21	100000	47.5 : 52.5	2494	2.5	83
22	100000	51.2 : 48.8	1159	1.2	46
23	100000	47.0 : 53.0	3037	3.0	40
24	100000	49.0 : 51.0	952	1.0	174
25	100000	50.0 : 50.0	48	0.0	212
26	100000	50.6 : 49.4	565	0.6	156
27	100000	51.8 : 48.2	1841	1.8	66
28	100000	51.6 : 48.4	1603	1.6	12
29	100000	49.9 : 50.1	83	0.1	188
30	100000	48.9 : 51.1	1133	1.1	46
31	1000000	51.2 : 48.8	12494	1.2	105
32	1000000	48.6 : 51.4	14424	1.4	53
33	1000000	49.8 : 50.2	2210	0.2	141
34	1000000	49.6 : 50.4	3572	0.4	37
35	1000000	50.9 : 49.1	8644	0.9	8
average		47.3 : 52.7		2.7	71

Tab. 12.4: Experimental Results for algorithm KnownSize.

13. CONCLUSION

This is the first deterministic result for non-trivially approximating splitters in streams storing only a constant number of reference elements. The median problem itself occurs in many algorithms and hence is quite important in theory and practice. Especially for tiny devices like sensor nodes in a network, the model is appropriate. Although several solutions exist using randomization and polylogarithmic memory, it is nevertheless of interest whether the same results can be achieved in a (simple) deterministic way and/or with less memory.

For streams with unknown size, the case for storage size two is solved asymptotically optimal. For arbitrary storage space an approximation algorithm is presented. A multi-pass solution for finding the exact median or any element of a specified rank is presented. Several results are not only asymptotic, but are given explicitly with very reasonable constants.

All the algorithms are fast and simple and have a tiny memory footprint in practice. They have worst-case guarantees but showed a much better average behavior as shown for random streams in Chapter 12.

Whether a linear approximation is achievable with constant storage or not remains an interesting open problem.

BIBLIOGRAPHY

[ABE98] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60:125–135, 1998.

[Alt01] Ernst Althaus. *Curve Reconstruction and the Traveling Salesman Problem*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2001.

[AM00] E. Althaus and K. Mehlhorn. Polynomial time TSP-based curve reconstruction. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 686–695, January 2000.

[Bei90] Richard Beigel. Unbounded searching algorithms. *SIAM Journal on Computing*, 19(3):522–537, 1990.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[Bes95] Sergei N. Bespamyatnikh. An optimal algorithm for closest pair maintenance. In *Proc. 11th Annu. Sympos. Comput. Geom.*, pages 152–161, 1995.

[BFP+73] Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

[Bru05] F. Thomas Bruss. What is known about Robbins' problem? *Journal of Applied Probability*, 42:102–120, 2005.

[BY76] Jon Louis Bentley and Andrew Chi-Chih Yao. An almost optimal algorithm for unbounded searching. *Information Processing Letters*, 5:82–87, 1976.

[CMRS64] Y. S. Chow, S. Moriguti, H. Robbins, and S. M. Samuels. Optimal selection based on relative rank (the secretary problem). *Israel Journal of Mathematics*, 2:81–90, 1964.

[CSD02] D. Cohen-Steiner and F. Da. A greedy delaunay based surface reconstruction algorithm. Rapport de recherche 4564, INRIA, 2002.

[DK99] T. K. Dey and P. Kumar. A simple provable algorithm for curve reconstruction. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 893–894, January 1999.

[DMR00] T. K. Dey, K. Mehlhorn, and E. A. Ramos. Curve reconstruction: Connecting dots with good reason. *Comput. Geom. Theory Appl.*, 15:229–244, 2000.

[DW01] T. K. Dey and R. Wenger. Reconstructing curves with sharp corners. *Comput. Geom. Theory Appl.*, 19:89–99, 2001.

[DW02] T. K. Dey and R. Wenger. Fast reconstruction of curves with sharp corners. *Int. J. Comput. Geometry Appl.*, 12(5):353–400, 2002.

[EKS83] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, IT-29:551–559, 1983.

[Eri03] Jeff Erickson. Nice point sets can have nasty delaunay triangulations. *Discrete & Computational Geometry*, 30(1):109–132, 2003.

[FR01] Stefan Funke and Edgar 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.

[Fre02] Daniel Freedman. Combinatorial curve reconstruction in hilbert spaces: A new sampling theory and an old result revisited. *Comput. Geom.*, 23(2):227–241, 2002.

[Fun01] Stefan Funke. *Combinatorial Curve Reconstruction and the Efficient Exact Implementation of Geometric Algorithms.* PhD thesis, Universität des Saarlandes, Postfach 151150, D-66041 Saarbrücken, Germany, 2001.

[Gie99a] J. Giesen. Curve reconstruction, the TSP, and Menger's theorem on length. In *Proc. 15th Annu. Sympos. Comput. Geom.*, pages 207–216, 1999.

[Gie99b] Joachim 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.

[GK01] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the*

2001 ACM SIGMOD international conference on Management of data, pages 58–66, New York, NY, USA, 2001. ACM Press.

[GM06] Sudipto Guha and Andrew McGregor. Approximating quantiles and the order of the stream. In *25th Symposium on Principles of Database Systems*, pages 273–279, 2006.

[GMV06] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *17th SIAM-ACM Symposium on Discrete Algorithms*, pages 733–742, 2006.

[GS01] C. Gold and J. Snoeyink. A one-step crust and skeleton extraction algorithm. *Algorithmica*, 30:144–163, 2001.

[GW04] Paul Goodwin and George Wright. *Decision Analysis for Management Judgment*. John Wiley & Sons, 2004.

[JC85] Raj Jain and Imrich Chlamtac. The p2 algorithm for dynamic calculation of quantiles and histograms without storing observations. *Communications of the ACM*, 28(10):1076–1085, 1985.

[KRG] Ulrich Kortenkamp and Jürgen Richter-Gebert. Cinderella. `http://www.cinderella.de`.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, 1987.

[Len05a] Tobias Lenz. Reconstructing collections of arbitrary curves. In *Proceedings of the 21st Annual Symposium on Computational Geometry*, pages 366–367, Pisa, Italy, June 2005.

[Len05b] Tobias Lenz. Simple reconstruction of non-simple curves. Technical Report B 05-02, Freie Universität Berlin, March 2005.

[Len06a] Tobias Lenz. Deterministic splitter finding in a stream with constant storage and guarantees. In *17th International Symposium on Algorithms and Computation*, volume 4317 of *Lecture Notes in Computer Science*, pages 26–35, Kolkata, India, December 2006. Springer.

[Len06b] Tobias Lenz. How to sample and reconstruct curves with unusual features. In *22th European Workshop on Computational Geometry*, pages 29–32, Delphi, Greece, March 2006.

[LZJ+05] DanFeng Lu, HongKai Zhao, Ming Jiang, ShuLin Zhou, and Tie Zhou. A surface reconstruction method for highly noisy point clouds. In *Workshop on Variational, Geometric & Level Set Methods (VLSM)*, volume 3752 of *Lecture Notes in Computer Science*, pages 283–294. Springer, 2005.

[Mat93a] J. Matoušek. Geometric range searching. Tech. Report B-93-09, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 1993.

[Mat93b] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993.

[MLT00] Gerard Medioni, Mi-Suen Lee, and Chi-Keung Tang. *Computational Framework for Segmentation and Grouping*. Elsevier Science Inc., 2000.

[MP80] J. Ian Munro and Mike Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.

[MR96] J. Ian Munro and Venkatesh Raman. Selection from read-only memory and sorting with minimum data movement. *Theoretical Computer Science*, 165(2):311–323, 1996.

[MRL98] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. *SIGMOD Rec.*, 27(2):426–435, 1998.

[Pat96] Mike Paterson. Progress in selection. In *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory*, volume 1097 of *Lecture Notes In Computer Science*, pages 368–379, 1996.

[RRR98] Monika Rauch Henzinger, Prabhakar Raghavan, and Sridar Rajagopalan. Computing on data streams. Technical Note 1998-011, digital Systems Research Center, Palo Alto, CA, 1998.

[Vit85] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

INDEX