FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Advancing-Front Mesh Resampling to Local Feature Size

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Lukas Geyer
Matrikelnummer 1026408

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Mag. rer. soc. oec. PhD Stefan Ohrhallinger

Wien, 5. Dezember 2014

Lukas Geyer   Michael Wimmer

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Advancing-Front Mesh Resampling to Local Feature Size

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Lukas Geyer
Registration Number 1026408

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Mag. rer. soc. oec. PhD Stefan Ohrhallinger

Vienna, 5th December, 2014

Lukas Geyer	Michael Wimmer

Erklärung zur Verfassung der Arbeit

Lukas Geyer
Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 5. Dezember 2014

Lukas Geyer

Kurzfassung

Diese Bachelorarbeit beschreibt einen neuen Algorithmus für das sequentielle Resampling eines Polygons. Weiters wird der Algorithmus angepasst um auch mit verketteten quadratischen Bézier-Kurven zu arbeiten. In beiden Fällen erzeugt der Algorithmus ein Polygon das eine Annäherung an die Eingabe darstellt. Der Algorithmus wurde so konstruiert dass die Verbindungen zwischen den Ergebnispolygon-Punkten durch einen weiteren Algorithmus wiederhergestellt werden können; dieser Algorithmus wird ebenfalls in dieser Arbeit beschrieben. Diese Eigenschaft erlaubt das Weglassen der Verbindungen – wird nur die Punktwolke gespeichert stellt das keinen Informationsverlust dar. Ebenfalls in dieser Arbeit enthalten sind Ansätze den Algorithmus auch auf dreidimensionale Meshes anzuwenden.

Abstract

This thesis describes a new algorithm for the sequential resampling of a polygon. Furthermore the algorithm is adapted to work with a chain of quadratic Bézier-curves. In both cases the algorithm produces a polygon approximating the original input. The algorithm is designed in a way that the connections between the points of the resulting polygon can be reconstructed by another algorithm that is also described in this thesis. This allows to neglect the connections and only store the point cloud without information loss. The thesis also describes attempts to adapt the algorithm to work with three-dimensional meshes.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Two-dimensional polygons and three-dimensional meshes often contain more samples than needed for a satisfying representation. In these cases, resampling helps reducing the amount of samples, either to decrease the needed storage space or to reduce the data that has to be processed when working on these polygons/meshes. The algorithm described in this thesis is intended to do this resampling.

There are already other resampling algorithms, but many of them start with the original mesh and remove some points while moving the other ones around to reach simpler approximations. The algorithm in this thesis instead gradually builds the resampled polygon/mesh with the goal to adapt to local features more precisely. The other goal of this algorithm is that the results of the resampling process shall support edge retrieveability: the connections do not have to be stored – a simple algorithm (also described in this thesis) can reconstruct the correct connections out of the point cloud. This allows additional storage space savings.

1.2 Related work

One of the already existing methods to simplify a very complex mesh is vertex clustering [RB93]; it has the advantage that it gets along with low computational effort. On the downside, vertex clustering might cause topology changes. The mesh can become non-manifold and regions with higher detail are not considered explicitly.

When using the edge collapse technique [Hop96] extended optimization is impeded by the fact that the mesh is not built new. The vertices are more likely to stay in the same location. In addition to this, edge collapsing can lead to self-intersections in the mesh.

The algorithm described in this thesis has significantly higher computational effort than the two aforementioned approaches, but in return it has the ability to locally adapt

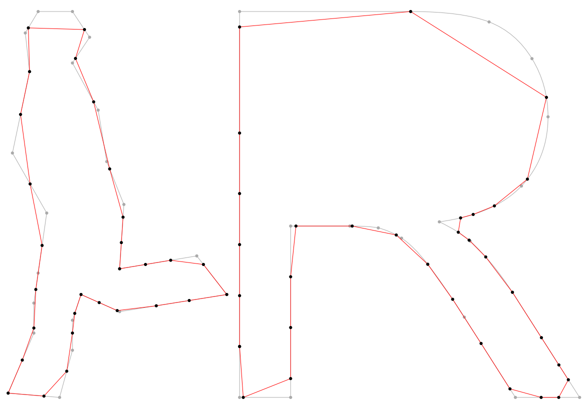

Figure 1.1: An example of a polygon and a path of Bézier-curves resampled with this algorithm (with deviation restriction deactivated)

to the degree of detail in the mesh and supports edge retrieveability as described in section 2.1 on the facing page.

CHAPTER 2

Resampling in 2d: polygons

2.1 Vocabulary definitions

A list of problem specific words used in this chapter. For descriptions of the used conditions and methods see section 2.4 on page 5.

edge In this chapter, *edge* refers to a straight line segment with a start- and an endpoint, that is part of a polygon.

polygon Every mention of *polygon* in this thesis refers to the closed path of edges, not the area enclosed in it.

vertex Every occurrence of the word *vertex* in this chapter refers to the point where two edges meet.

input polygon/original polygon The polygon that was passed to the algorithm.

output polygon/resampled polygon/resampling result The polygon that the algorithm creates when applied to the input polygon.

resampling The process of approximating a (subsection of a) polygon with another polygon.

edge retrieveability This property means that if only the vertices of a polygon are given, its edges (i.e. the correct connections) can be reconstructed.

deviation This value is used to measure the divergence of the output polygon from the input polygon. For a detailed description how it is measured see section 2.4.3 on page 6.

sharp corner This refers to a vertex where the two edges enclose an angle smaller than $90\,°$.

corner chamfering In some cases during resampling the algorithm needs to cut off sharp corners. For more information about corner chamfering, see section 2.5.1 on page 9.

leading away/monotonically increasing distance [from a point p] Describes a property of a sequence of edges. It is fulfilled if a point $p_{wandering}$ starting at the edge's beginning p_{start} has, while following the edge sequence till its end, monotonically increasing distance to p. For details about its evaluation see *About the coherence with* $90\,^\circ$ *angles* in section 2.4.1.

2.2 Overview of the algorithm's purpose

The purpose of the algorithm is to resample an input polygon with as few vertices as possible while maintaining a given upper limit for the deviation. The algorithm produces a new polygon as result. The vertices of the resampling result lie on the input polygon. Due to this property, the algorithm may be affected by area shrinking –that means that the output polygon encloses a smaller area than the input polygon. If the resampling algorithm is repeatedly called, each time upon the resampling result from the previous algorithm call, the area might continue shrinking, leading to further deviation and deformation from the original polygon.

As additional goal, the algorithm should produce output polygons that support edge retrieveability.

The algorithm can be adapted to one's needs with a set of parameters. With the parameter *maximum_deviation* it is possible to define the upper limit for the deviation (for a detailed description of the meaning of the passed value see section 2.4.3 on page 6). The parameter *chamfering_precision* controls the maximum allowed distance to cut off during corner chamfering (also see section 2.5.1 on page 9). The parameter *discretion_precision* is used to set the desired precision for the bisection method (see section 2.4.4 on page 7). With the parameter *reconstruction_condition* one of the conditions needed to guarantee edge retrieveability can be deactivated.

The parameters *chamfering_precision* and *discretion_precision* are defined in terms of the diagonal of the polygon's bounding box.

2.3 The edge reconstruction algorithm

The algorithm that can be used to retrieve the correct edges is quite simple: every vertex should have two edges; the first connection is created with the nearest neighbor. To avoid edges to vertices that might better get connected to the nearest neighbor and should therefore not be regarded as options, the algorithm rejects all vertices that are nearer to the neighbor than to the currently regarded sample. Out of the remaining possible vertices, the nearest one is chosen for the second edge (see Figure 2.1). For a related algorithm, see [OM13].

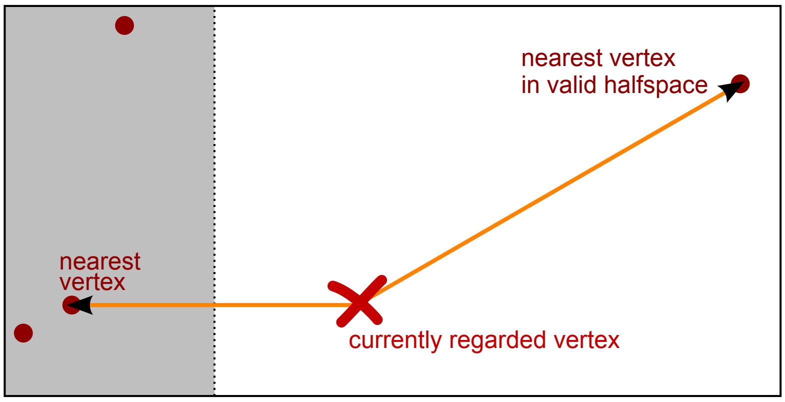

Figure 2.1: The reconstruction algorithm chooses the nearest vertex to the currently regarded vertex. All vertices in the gray halfspace are closer to the nearest vertex than to the currently regarded vertex and therefore not taken into consideration for the second connection.

2.4 Overview of the used conditions and methods

The resampling algorithm uses three conditions respectively restrictions. In the following these conditions are shortly described for the two dimensional version of the algorithm.

2.4.1 Homeomorphic condition

The homeomorphic condition checks inside a circle with its center lying on the polygon for a property described in the following lines. The homeomorphic condition is fulfilled for a radius r and center c, if for every circle a with center c and $radius \leq r$, the following assertion is applying: The subsection of the polygon that lies inside a is a single connected component.

To describe this property in another way that is more suitable for implementation purposes: the circle center c is a point lying on the polygon; the function $p(t)$ describes the polygon, where $p(0) = c$. When t is increasing from 0, $p(t)$ returns the path along the polygon from c in one direction, with decreasing t it follows the polygon in the other direction. At first, $p(t)$ obviously diverges from $p(0)$ when t increases as well as when it decreases from 0. t is increased/decreased until $p(t)$ reaches the circle. If the distance from $p(t)$ to the start point $p(0) = c$ never decreased for both increasing and decreasing t, and the section of the polygon that was not traversed by $p(t)$ in one of the two runs does not intersect with the circle, the homeomorphic condition is fulfilled (see Figure 2.2).

About the coherence with 90° angles In this thesis, an edge is called *leading away from the center c* if a wandering imaginary point $p_{wandering}$ traverses the whole edge with continually increasing distance to c. An edge can either completely fulfill that condition or fail right at the start of the edge; it is not possible that the edge begins by leading

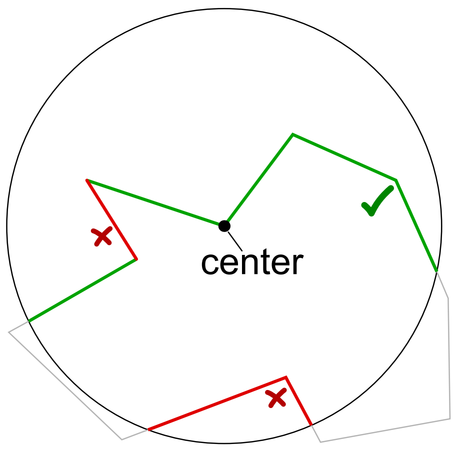

Figure 2.2: Example of a case where the homeomorphic condition is not fulfilled because of the red edges

away from c but later comes back closer, since the edges are straight. An edge leads away from c exactly then if a line from the center to the edge start encloses an angle of $90\,^\circ$ or larger with the edge (see Figure 2.3).

The reason of this condition For the edge reconstruction (as described in section 2.3 on page 4) to work correctly, the connection to the nearest sample to c must be a valid edge. If a point s_n inside the circle a is chosen for the next sample, then every point p beyond s_n is further away from c, so s_n is the nearest sample in the resampling direction. If the previous sample s_p is inside the circle a too, then it is the nearest sample opposite to the resampling direction. So the homeomorphic condition must also be applied in the reverse direction for a new sample. Since in this reverse direction the samples are already placed there might be the opportunity for some optimization, but this was not further elaborated in this thesis.

2.4.2 Reconstruction condition

This condition needs the coordinates of the previous sample s_p, the current sample s_c and the next sample s_n. The condition checks the following two inequalities: $|\overrightarrow{s_p s_n}| > |\overrightarrow{s_p s_c}|$ and $|\overrightarrow{s_p s_n}| > |\overrightarrow{s_c s_n}|$. That guarantees that s_c will not be disconnected during edge reconstruction (see 2.3 on page 4) by connecting s_p with s_n, like it happened in Figure 2.4.

2.4.3 Deviation restriction

When using only the homeomorphic condition and the reconstruction condition for resampling, the results show quite high deviations from the original polygon. The deviation restriction is dedicated to handle this problem.

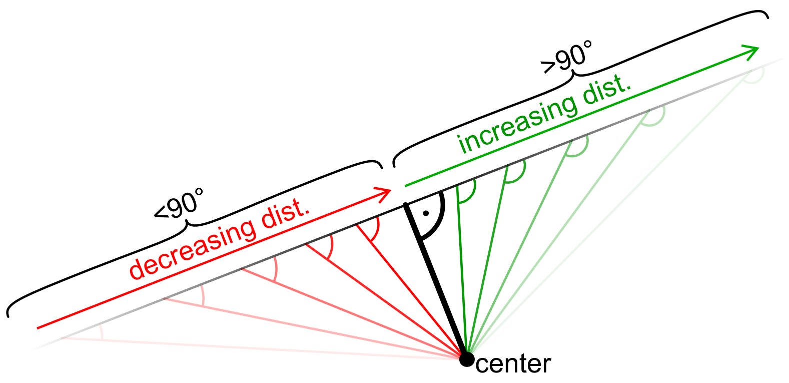

Figure 2.3: Illustration showing that angles below 90 ° lead to edges not continually leading away from the center; the start point of the edge is varying – if it lies in the red section the edge does not continually lead away from the center

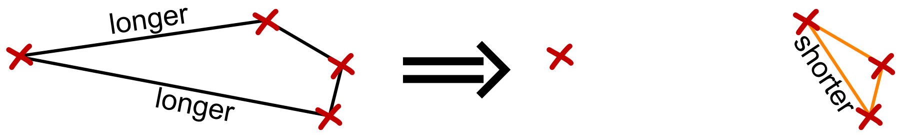

Figure 2.4: The edge reconstruction algorithm connects nearest neighbors, so if a sample's (in the following called *middle sample*) neighbor-samples are closer together than to the middle sample, the shorter connection will be chosen and the middle sample is excluded

The initial question is how to define the deviation. The approach for this algorithm was to look at every edge of the resampled polygon separately and define their deviation. The deviation of an edge e has been defined as the farthest distance between e and its corresponding original path. In the two-dimensional case, e can easily be matched with the original polygon subsection that was replaced by e. Every sample lies on the original polygon, so the obvious approach is that the original polygon's subsection from the start of e to its end corresponds to this sample edge (see Figure 2.5).

The deviation restriction is executed right after the homeomorphic condition and the reconstruction condition proposed a valid next sample with distance r. To restrict the deviation in the resampling result, the parameter *maximum_deviation* can be passed to the algorithm; the allowed deviation d is calculated in the following way: $d = maximum\_deviation \cdot r$, so that smaller features in the original polygon are resampled with more precision. d represents the target maximum deviation for e.

2.4.4 Bisection method

The bisection method is used to approximate a maximum value of a non-convex function that fulfills all given conditions. A maximum value is given; if the conditions are not fulfilled search in the interval from 0 to the maximum value: choose the value in the

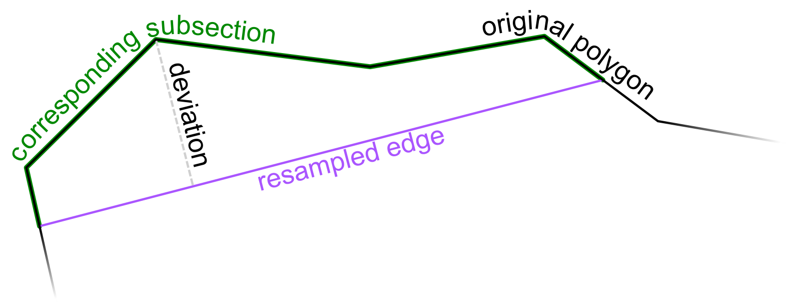

Figure 2.5: Definition of the deviation of a sample edge

middle of the interval and check again if the conditions apply. If they do, continue the search in the upper half of the interval, otherwise search in the lower half. Repeat this procedure until the desired precision is reached – this precision threshold can be passed to the algorithm as parameter *discretion_precision* [BF85].

2.5 Implementation decisions for the 2-dimensional version (polygon resampling)

From the beginning, a desired feature of the resampling results was the ability to reconstruct the connections of the sample points, so that the edges do not need to be stored (edge retrieveability). At first, the homeomorphic condition was considered to suffice for granting this ability.

So, the first approach consisted of the following steps (sharp corners neglected, see section 2.5.1 on the next page):

1. select an original vertex from the given polygon as the current sample
2. search for farthest possible sample from current sample with bisection method, using the homeomorphic condition
3. find the sample in between the current sample and the farthest possible sample that fulfills the deviation restriction
4. if the path from the current sample to the next sample contains the start vertex, the next sample is set to the start vertex. The resampled polygon is closed and is returned as the result. Otherwise, store the next sample, make it the current sample and continue with step 2.

When it was implemented and tested with a rectangle while disabling the deviation restriction, it became obvious that the homeomorphic condition is not sufficient (see Figure 2.6). The algorithm started with one of the four corners, and since the distance from the corner to two imaginary points following the polygon outline in the two directions is increasing till the opposite corner is reached by both points, this opposite corner is

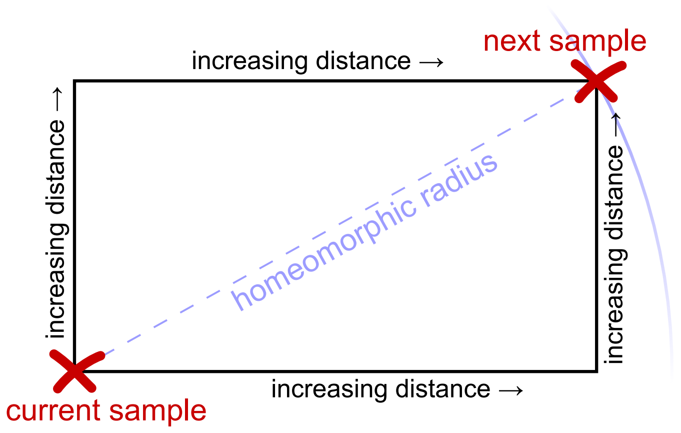

Figure 2.6: First resampling of a rectangle; the homeomorphic condition allows the opposite corner as next sample

selected as the next sample. This results in a similar situation as with the start vertex – again, the opposite corner becomes the next sample, the resulting resampled polygon is closed and the algorithm has finished. So the resampled rectangle consists only of two points; while this might be an accurate representation considering there are no detailed features in a rectangle, this result misses the desired edge retrieveability. As a consequence, the reconstruction condition was added to the checked conditions during bisection method in step 2, to assure the ability to retrieve the edges.

Since the homeomorphic condition guarantees that the points on the path continually grow in distance to the current sample, it is not obvious why the sample after the next sample can lie closer than the next sample; all points following the next sample should be farther away. The problem in this thought is, that there are two paths leading away from the current sample/homeomorphic circle center. So if the sample after the next sample lies on the other path it might be closer to the current sample than the next sample. If the homeomorphic circle contains (nearly) the whole polygon, this situation might occur; the sample after the next sample might already lie on the other path, as shown in Figure 2.7.

2.5.1 The problem with sharp corners

Intuitively, the homeomorphic condition requires that the path starting at the current sample does not turn around and come back closer to the current sample inside the homeomorphic radius (for a more precise description see section 2.4.1 on page 5). This implies that corners enclosing an angle smaller than 90° are unpassable obstacles. The edge after the sharp corner starts with decreasing distance to every point of the edge before the sharp corner (except for its end point); so if the current sample lies on the edge before the sharp corner, the homeomorphic condition makes it impossible to pass the sharp corner. The algorithm keeps decreasing the distance between the samples but

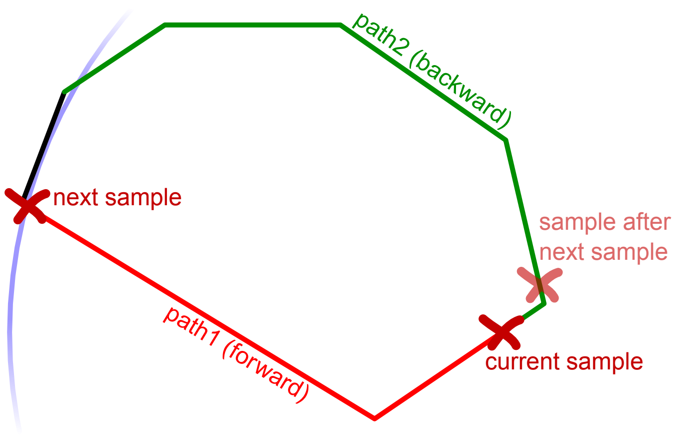

Figure 2.7: From the current sample, due to the homeomorphic condition, all points following the next sample lie further away from the current sample, except if the point lies on the other path (green).

never reaches the sharp corner, as shown in Figure 2.8.

To solve this behavior a sharp corner check for the original polygon's vertex that follows the current sample is performed right before using the bisection method. If a sharp corner has been detected, the problem is tried to be solved with special treatment, described in the following paragraphs. Only if this special treatment cannot be applied the next sample is computed with the bisection method as usual.

The problem can be separated into two different cases: the sharp corner is between 60° and 90°, or the sharp corner is between 0° and 60°.

Sharp corner between 60° and 90° The homeomorphic condition prohibits the resampling of these corners, but there exists a solution that fulfills the desired edge retrieveability and even is an exact representation of this polygon feature. So, this special case is handled by the algorithm in a separate computation routine, where the corner vertex is assumed as the next sample. The homeomorphic condition still has to be checked, but it is ignoring the edge of the input polygon immediately following the sharp corner.

To finish the special treatment of this sharp corner, the sample following to the next sample has to be computed and validated. The position of this new sample is computed by selecting the sample that forms an isosceles triangle together with current sample and next sample. That means, the distance between next sample and current sample is the same as between next sample and the following new sample (Figure 2.9).

Sharp corner between 0° and 60° The only solution for handling these corners while guaranteeing the retrieveability of connections as well as that the samples lie on the original polygon is to chamfer the problematic corner. This leads to a certain loss of detail in this specific polygon feature. Therefore the maximum distance between the

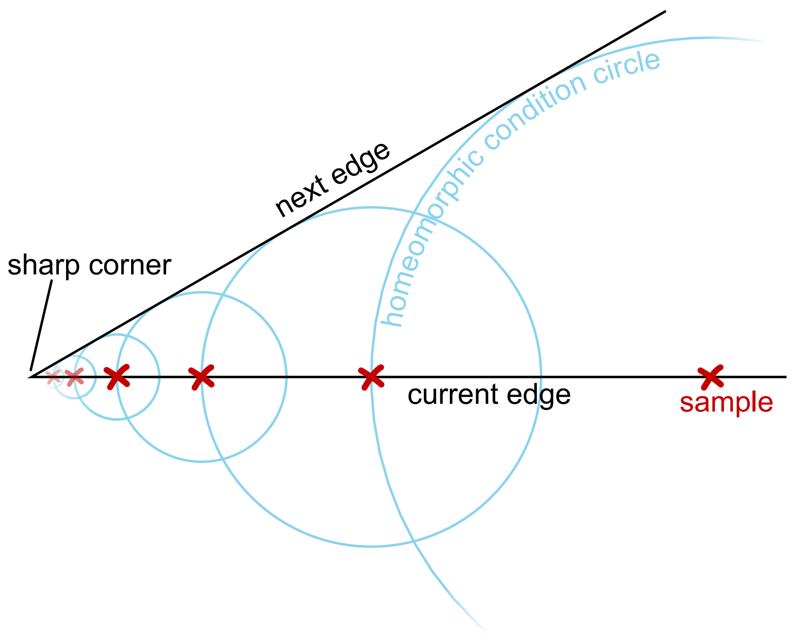

Figure 2.8: The homeomorphic condition prohibits the passing of sharp corners

current sample and the sharp corner that must be surpassed before chamfering is started can be passed as parameter to the algorithm.

As shown in Figure 2.10, the next sample is assumed to be at the same distance from the sharp corner as the current sample, but lying on the input polygon's next edge. For the homeomorphic condition check, the edge immediately following the sharp corner is ignored. If the condition is not fulfilled for the next sample, the chamfering cannot be done in this iteration; the next sample is dropped and instead computed in the default way.

2.5.2 Influence of sharp corners on the starting point of the algorithm

The reference implementation of the algorithm accepts as input p2d-files, which are simple text-files with a list of x- and y-coordinates of the vertices, separated by a blank, and each vertex in its own line. In initialization, the algorithm sets the current sample to the first sample listed in the input p2d-file. Since this vertex could be a sharp corner the sample cannot simply be transferred into the result polygon. Instead, the algorithm computes the next sample to the (potentially sharp) first vertex, which then becomes the start vertex of the result polygon. The first vertex from the input file, even though it was used as the first current sample, is not transferred to the result polygon.

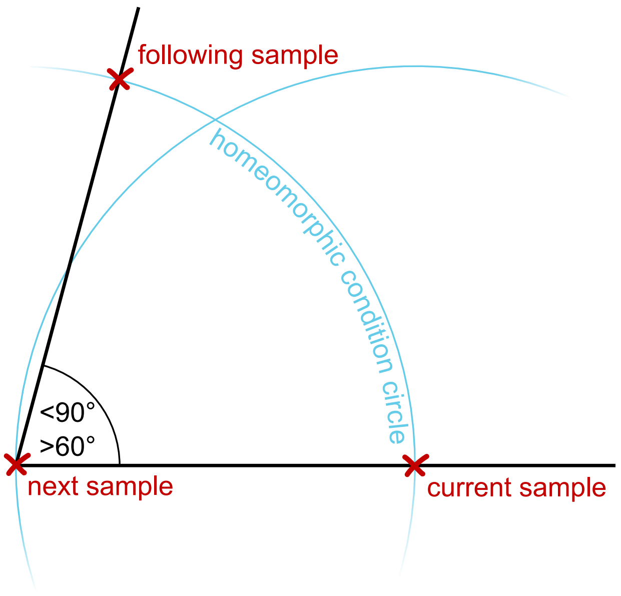

Figure 2.9: Special treatment of sharp corners over 60 °; the homeomorphic condition has to be checked for all 3 samples

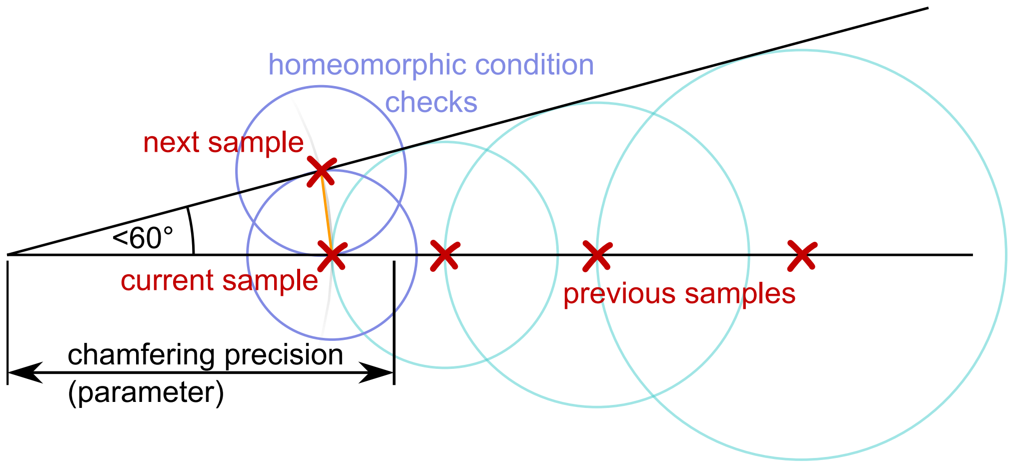
homeomorphic condition checks
next sample
<60°
current sample
previous samples
chamfering precision (parameter)

Figure 2.10: Special treatment of sharp corners under 60 °; the homeomorphic condition has to be checked for current and next sample

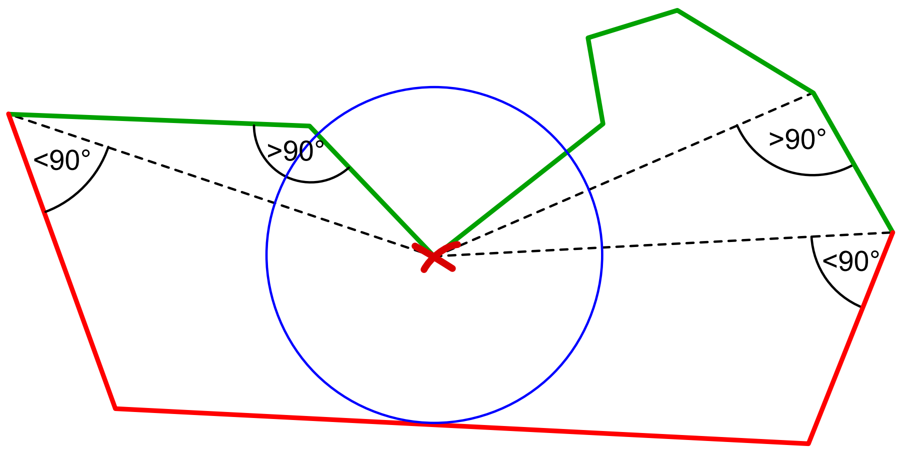

Figure 2.11: The homeomorphic algorithm separates into accepted edges (green) and remaining edges (red), to which the minimum distance is the homeomorphic radius (blue circle)

2.5.3 Algorithm for the homeomorphic condition

As described in section 2.4.1 on page 5 the homeomorphic condition ensures that inside a specific radius, the two paths starting from the circle center have continually increasing distance to the center of the circle. The first version of the algorithm for checking the homeomorphic condition needed in addition to the tested center a given radius. The algorithm could only determine if the condition is fulfilled in the given case or not. The greatest valid radius had to be approximated using the bisection method. While working on the 3d-version of the algorithm an improved version of it has been developed. The updated version just needs the center of the circle as parameter c, the largest possible radius r is computed for this point.

The radius r is computed with the following steps (see also Figure 2.11):

1. start at the edge of the original polygon on which c lies

2. follow the original polygon path in one direction and check for every edge if the angle between the edge and the line from c to the edge startpoint is at least $90\,°$; add these edges to the list of accepted edges

3. stop at the first edge that does not fulfill this condition (this edge is excluded from the accepted edges list)

4. do the same steps in the other direction and add the accepted edges to the list

5. compute the minimal distance from c to the edges of the original polygon that are *not* in the list; this is the resulting radius r

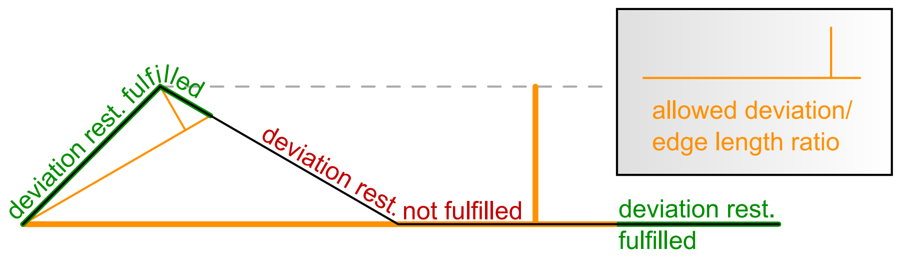

Figure 2.12: Two solution ranges when using scaling deviation threshold

2.5.4 Algorithm for the deviation restriction

At first, the deviation was planned to scale with the length of the resampled edge, but later the dependency changed to the length of the longest allowed edge using the homeomorphic and the reconstruction condition. The reason for this change was that with the dependency on the resampled edge's length there might be ambiguous solutions, because with growing edge length, smaller deviations that prohibited solutions at smaller edge length become acceptable, which leads to further solution ranges (see Figure 2.12). The subsequently developed algorithm makes use of the fact that the allowed deviation can be computed as absolute value for the currently considered edge rather than changing it according to the edge length. Multiple solution ranges can still occur, but it is possible to determine the limit after which no more valid solutions can occur.

The algorithm consecutively computes the valid angle range for the new edge. This range shrinks the further the next sample is assumed, until the range narrowed down to one solution. Since the deviation extrema are always the vertices of the original polygon only these need to be considered for the range restriction process.

For a graphical representation of the algorithm method, see Figure 2.13. Beginning with the vertex at the end of the original polygon's edge that contains the current sample, a circle with $radius = allowed\ deviation$ is constructed around the vertex. The angles of the two tangents laid from the current sample to the circle constrain the valid angle range. For the edge after the checked vertex, the farthest solution inside the valid range is computed. If there is no solution on this edge because it does not intersect with the valid angle the last found solution remains the best for this step. This is done as long as a valid angle range remains. There can not be solutions beyond the vertex that leads to no remaining range, so the last found solution has to be the best.

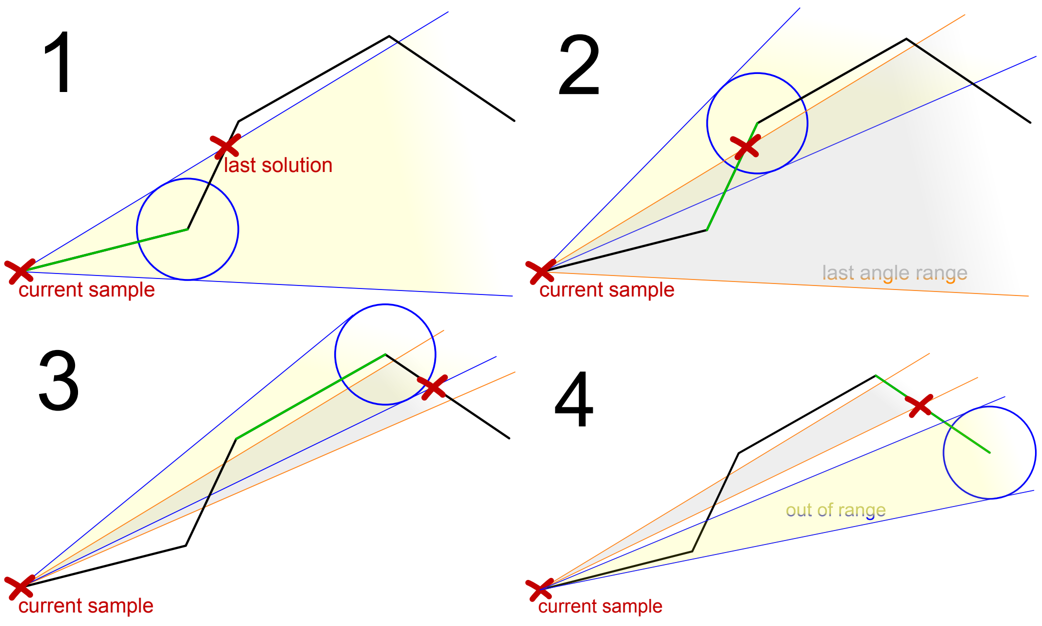

Figure 2.13: The steps of the deviation restriction algorithm

CHAPTER 3

Resampling in 2d: Bézier-curves

3.1 Vocabulary definitions

A list of problem specific words used in this chapter. The Bézier-version of this algorithm uses the same conditions as already described in the polygon-chapter in section 2.4 on page 5.

polygon, edge These words are used with the same meaning as in the chapter Resampling in 2d: polygons. See *Vocabulary definitions* in that chapter.

curve This refers to a quadratic Bézier-curve. The three control points are named: start point, control point and end point. That means every following mentioning of a *control point* refers to the second/middle control point.

control line segment In this chapter, this refers to the following two edges of the Bézier-control-polygon for a curve: *start point–control point* and edge *control point–end point*

vertex Every occurrence of the word *vertex* in this chapter refers to the point where two edges or two curves are connected.

path In this chapter, *path* refers to a chain of curves.

input path The closed path that was passed to the algorithm.

output polygon/resampled polygon/resampling result The polygon that the algorithm creates when applied to the input path.

resampling The process of approximating a (subsection of a) path with a sequence of edges.

deviation This value is used to measure the divergence of the output polygon from the input path. It is measured in the same way as in the polygon version – for a detailed description see section 2.4.3 on page 6.

sharp corner This refers to a vertex where the tangents of the two meeting curves enclose an angle smaller than 90°.

corner chamfering In some cases during resampling the algorithm needs to cut off sharp corners. For more information about corner chamfering, see section 3.3.1 on the facing page.

leading away/monotonically increasing distance [from a point p] Describes a property of a curve. It is fulfilled if a point $p_{wandering}$ starting at the curve's beginning p_{start} has, while following the curve till its end, monotonically increasing distance to p. For details about its evaluation see section 3.3.2 on the next page.

split a curve The splitting of a curve is done with help of the De Casteljau algorithm [BF84], with parameter 0.5.

3.2 Overview of the algorithm's adaption for Bézier-curves

After the algorithm was working for resampling of polygons, the next goal was to extend it to work with other input types. While still staying in the 2-dimensional space, the connections between the input vertices are no longer straight edges, but quadratic Bézier curves. The resampling results are still polygons.

The input path must not intersect with itself, this also implies that there must not exist two equal subpaths.

The purpose of the algorithm and the resampling result's properties stay the same as described in section 2.2 on page 4. The parameters described in that section are also available in this version of the algorithm. It uses the same conditions as the polygon version, but the steps for their evaluation partially had to be adapted. It was not necessary to alter the reconstruction condition because it only regards the output polygon.

3.3 Implementation decisions for the 2-dimensional version (Bézier-curve resampling)

Quadratic Bézier-curves can show some properties that might complicate some tasks needed for this algorithm: the distance between a point s lying on the curve and a point p that starts at s and follows the curve's route might begin to decrease at some point, meaning that the curve has no monotonically increasing distance from s (see Figure 3.1). So the curve might approach itself. To eliminate these cases the input path is preprocessed, where all curves that might not have monotonically increasing distance are split up into two curve segments. This is repeated until all curves are accepted.

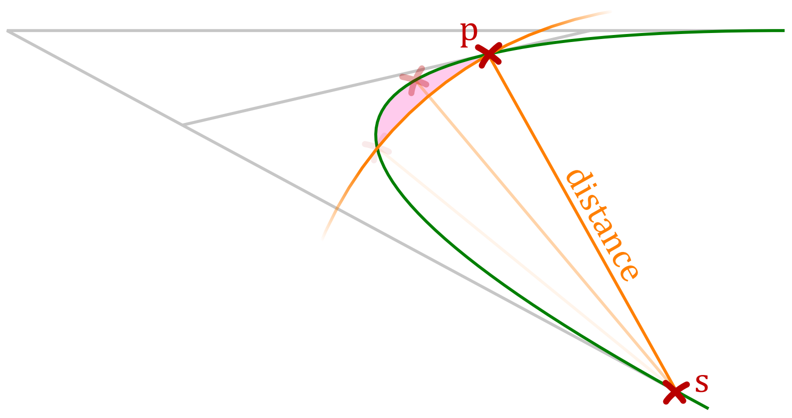

Figure 3.1: The distance between p and s might decrease at some point while p is following the curve.

The heuristic to classify the curve works in the following way: if the two control line segments enclose an angle smaller than $90°$ the curve is classified as *potentially lacking monotonically increasing distance*.

Since subsequent partitioning of a curve approximates to a line, and lines always have monotonically increasing distance, the preprocessing is guaranteed to terminate.

As a result of this preprocessing, it is easy to determine the sequential order of multiple given points lying on a curve: the points can be sorted according to their distance to the curve's start point.

3.3.1 Sharp corners between curves

Sharp corners can not occur within a curve, except if all three control points lie on a line and the middle control point does not lie between start and end point – but since this case is not accepted as valid input path, it can be neglected. This means that only the connections between the curves have to be regarded for sharp corner treatment.

The angle between two curves can be computed using the tangent lines at the end point of the first curve and the start point of the second curve. These tangents are already explicitly given with the control polygons of the curves.

After the angle has been calculated, sharp corners can be handled in exactly the same way as in the case of polygon resampling (see section 2.5.1 on page 9).

3.3.2 Algorithm for the homeomorphic condition

The condition is the same as used for polygons (see section 2.4.1 on page 5). As with the polygon version, the algorithm for the homeomorphic condition check was updated to the better version developed during the work on the algorithm's 3d-version, that is now able to compute the largest radius still fulfilling the condition instead of just evaluating it for a certain radius.

The algorithm gets a point c lying on the input path as parameter – the circle center. For this point c, the largest valid radius r is computed and returned.

The steps of the algorithm are mostly equivalent to its polygon version (see section 2.5.3 on page 13). The differences are how it is determined if a curve has monotonically increasing distance to c and how the distance of a curve to c is computed. In the polygon version, the elements between the vertices were straight edges; for these elements the 90° check was sufficient. For the Bézier-curves, a recursive approach was used.

The steps for the determination if a curve has monotonically increasing distance to c are (compare to Figure 3.2):

1. Check if the first control line segment of the curve leads away from c (with the 90° check described in section 2.4.1 on page 5). If not, the curve does not either, since the first control line segment is also the tangent of the curve at the start.
2. Check if the second control line segment leads towards c or in other words, if the line segment with swapped start and end point leads away from c. If it does, the curve does not have monotonically increasing distance, since the second control line segment is also a tangent of the curve.
3. Check if the second control line segment leads away from c. If it does, and the first control line segment leads away too (already checked in step 1), the whole curve leads away from c.
4. If no decision could be made yet, split the curve and check for both segments in a recursive way if they both have monotonically increasing distance. If one of the curve's segments does not fulfill this property then the curve itself does not either.

To compute the largest valid radius it is required to calculate the distances from c to every curve that is not in the list of accepted curves (compare with the polygon version).

The steps for the point – curve distance computation are:

1. If the distance between curve start and curve end is smaller than determined by the parameter *discretion_precision*, the shortest distance from c to one of the curve's three control points is returned.
2. For the next three tests, the curve's bounding triangle, constructed with the three control points, is regarded (see Figure 3.3). The side between start point and control point is called $side_1$, $side_2$ is between control point and end point, and the connection between start point and end point is called $side_3$. If one of the three tests applies, the nearest point is determined; the subsequent calculation of the distance between point–point is trivial.
   a) If $side_1$ and $side_3$ lead away from c, the start point is the nearest point.
   b) If $side_2$ and $side_3$ lead towards c, the end point is the nearest point.
   c) If $side_1$ leads away and $side_2$ leads towards c, either the start or the end point is the nearest point.

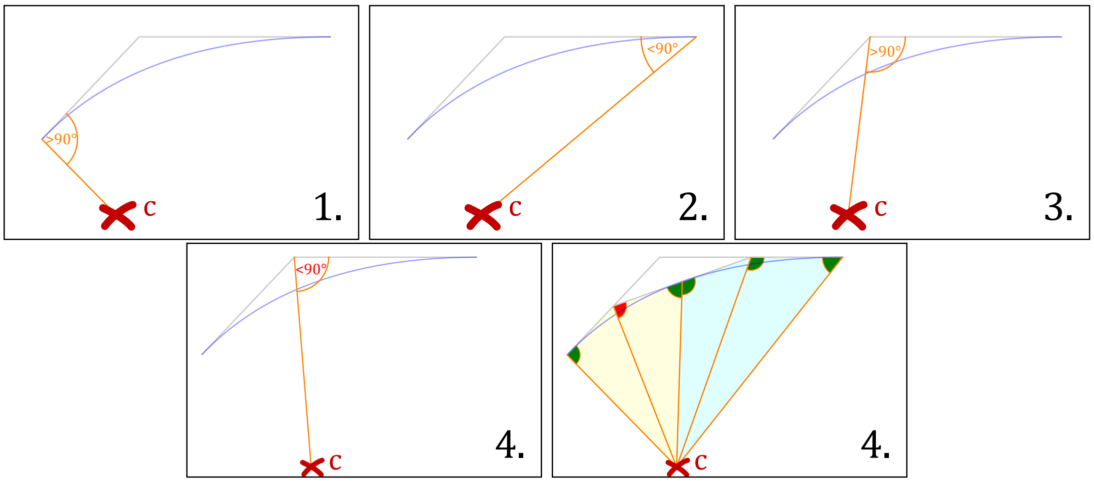

Figure 3.2: The angles checked during homeomorphic condition computation

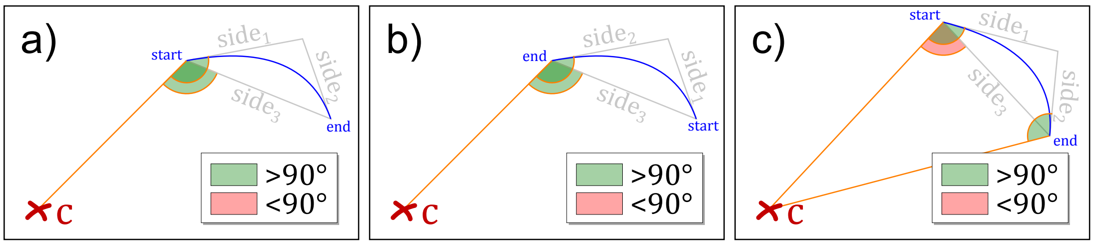

Figure 3.3: The checks performed during point–curve distance computation

3. If none of these conditions applied, the curve is split up and the steps are called for both segments in a recursive manner. The shorter resulting distance is returned as distance to this curve.

3.3.3 Algorithm for the deviation restriction

The core principle of the deviation algorithm stayed the same as in the polygon version. The difference is that the tangents laid form the current sample onto an edge always contain at least one of the two end points of the edge, whereas this does not apply for tangents laid onto a curve. This means that it is not sufficient to restrict the valid angle range at every vertex between the curves. The calculation of the tangents onto the curve is solved in a recursive manner.

The following steps describe the recursive algorithm that is used to find the farthest point on the path fulfilling the deviation restriction. In these steps, the valid angle range a will be restricted using the three control points of the curve; this is done by constructing a circle c with one of the three control points as center and the allowed deviation as radius. The tangents laid from the current sample *s* onto the circle *c* enclose an angle range, which is then intersected with *a* (see Figure 3.4).

1. Check if there is an angle range left after the angle restriction using the curve's end point.
2. If the distance from start point to end point is smaller than determined by the *discretion_precision* the recursion is stopped and the adapted angle range is returned; if after the restriction in step one no angle range is left, the start point of the curve is returned as the farthest valid point.
3. If there is angle range left it is possible that no further recursion is required. This is the case if the use of the control point as circle center would lead to no further angle restriction. If this applies, the adapted angle range is returned.
4. In any other case, further recursion is required; the curve is split up and these steps are called for each of the two parts (the second part uses the angle range returned by the first part, so they must be called in order).
5. The resulting angle range is returned, or if one of the parts led to a result, this is passed upwards in the recursion hierarchy.

The check in the first step determines if the farthest valid point might lie beyond this segment or if it is guaranteed to be somewhere in this curve part. The second step prevents endless recursion and stops if a certain precision is reached; the result from step one is used to return an adequate result.

Step three requires a more detailed explanation: Both angle range boundaries for a triangle are each most restricted at one of the three corners (see Figure 3.5). So it is sufficient to restrict the angle range using the three corners – no point inside the triangle would lead to further restriction. At least one of the corners does not contribute to the restriction, so it can be neglected (in the figure this is corner A); the angle range can be computed with just the other two corners. The triangle is now the curve's control polygon. If the control point that is not part of the curve leads to no further angle restriction then also all points that are inside the control triangle (and therefore also the points of the curve) would lead to no further angle restriction. So if there is angle range left (step one), this range is the result for the whole curve part.

Step four and five provide recursion functionality.

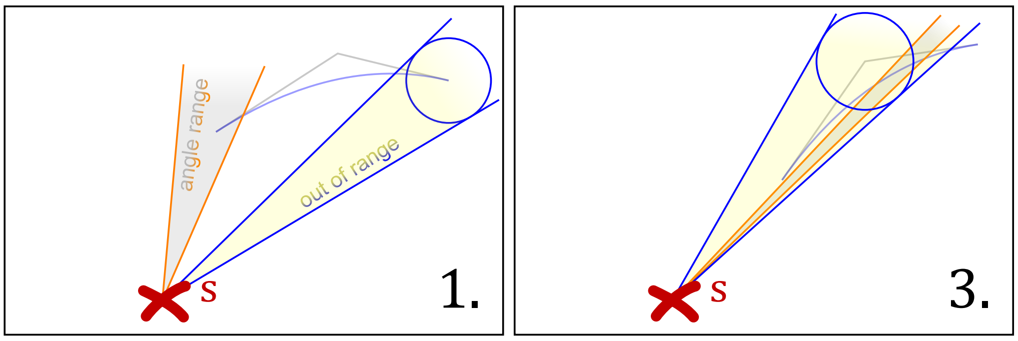

Figure 3.4: Illustration of the first and the third check for the deviation restriction. In the depicted case for the third check, the angle range a is entirely contained in the angle range between the tangents to the circle around the curves control point – the angle range a would not be further restricted and no recursion is required for this curve segment.

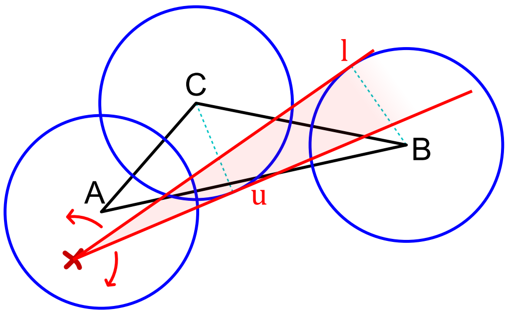

Figure 3.5: The angle range for a triangle ABC; the lower boundary l is most restricted by corner B, the upper boundary u by corner C

CHAPTER 4

Resampling in 3d

4.1 Vocabulary definitions

vertex A point with a position defined by three coordinates.

edge A connection line segment from one vertex to another one.

mesh *mesh* refers to a mesh consisting of vertices, edges and triangles. It must not contain holes and every edge is included in exactly two triangles. The mesh must not intersect with itself.

input mesh/original mesh The mesh that was passed to the algorithm.

output mesh/resampled mesh/resampling result The mesh that the algorithm creates when applied to the input mesh.

original triangle/edge/vertex A triangle/edge/vertex of the input mesh.

resampled triangle/edge/vertex A triangle/edge/vertex that is a (temporary) result in the resampling process.

resampling The process of approximating a (subsection of a) mesh with another mesh.

edge retrieveability This property means that if only the vertices of a mesh are given, its edges and surface triangles can be reconstructed.

deviation This value is used to measure the divergence of the output mesh from the input mesh. For more information about its measurement see section 4.3.3 on page 27.

sharp edge If the planes on which two triangles sharing one edge e lie enclose an angle smaller than $90\,^\circ$, the edge e is called a *sharp edge*.

sharp corner A vertex is called *sharp corner* if for its adjacent triangles $f_{0\ldots m}$ and edges $e_{0\ldots n}$ applies: there exists a triangle f_i which encloses with every edge $e_j, j = 0 \ldots n$ an angle smaller than $90\,^\circ$.

corner chamfering In some cases during resampling, the algorithm would need to cut off sharp edges/sharp corners. For more information about corner chamfering, see section 4.4.1 on page 31.

path with increasing distance [from a point p] This phrase is used to describe a property that is needed in the homeomorphic condition. It is fulfilled if a point $p_{wandering}$ starting at the path's beginning p_{start} has, while following the path till its end, monotonically increasing distance to p. If the point p is not explicitly mentioned, p_{start} is meant.

4.2 Overview of the algorithm in the three-dimensional space

The adaption of the algorithm to a three-dimensional version was not successfully finished because of many problems and special cases that emerged and needed sophisticated treatment. Further work on this problem would have exceeded the scope of this thesis and was therefore canceled. The final status on this topic will be described in this chapter.

The implementation of this algorithm relies on two libraries: vcg [otINRCI] and nl-opt [Joh].

The three-dimensional resampling algorithm is based on the two-dimensional version, and has therefore the same properties: vertices lie on original mesh, volume shrinking and edge retrieveability.

The parameters are the same as described in section 2.2 on page 4. However, *discretion_precision* was removed because the 3d-version does not use the bisection method.

4.3 Overview of the used conditions and methods

The core principle of the algorithm stays the same in its three-dimensional version; the same conditions and methods as in the two-dimensional version were used. Nevertheless, they had to be adapted to work with three dimensions. These adaptions are described in detail in the following sections.

4.3.1 Homeomorphic condition

The extension of this condition into three dimensions lead to the following definition:

The homeomorphic condition is fulfilled for a point c and a radius r if for every sphere s with center c and $radius \leq r$, the following assertion is applying: The subsection of the mesh that lies inside s is a single connected component.

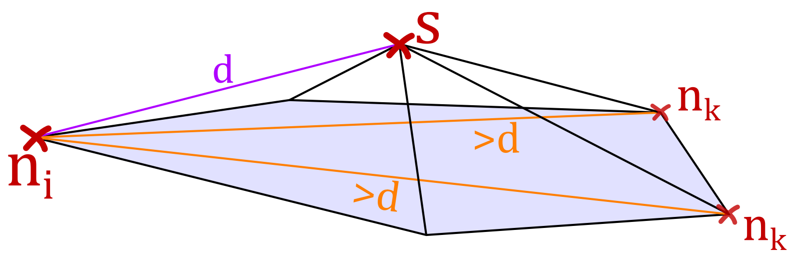

Figure 4.1: The reconstruction condition check for an umbrella: $|\overrightarrow{n_i n_k}| > |\overrightarrow{n_i s}|$

The definition that is more correlated to the used algorithm changed to the following:

The sphere center c lies on the mesh. The homeomorphic condition is fulfilled for a point c and a radius r if for every point p lying on the mesh where $|\overrightarrow{cp}| \leq r$, a path on the mesh from c to p with increasing distance can be found.

4.3.2 Reconstruction condition

The reconstruction condition still uses the neighbor samples of a resampled vertex, but other than in two dimensions there are now more than two neighbors. The sample s with its neighbors $n_{0...l}$ form an umbrella. For every sample $n_i; 1 \leq i \leq l$ must apply: its distance to every sample n_k that n_i does not share an edge with must be greater than its distance to s (see Figure 4.1). That guarantees that the connection between n_i and s will be reconstructed properly – otherwise the shorter connection might be preferred which leads to false edges.

In the two-dimensional case it was sufficient to consider one neighbor area, because the algorithm was sequential and only the previous samples had to be taken into account. In three dimensions, if a new connection (an edge) is established between two points, both points might already have other neighbors and the newly created edge might have influence on the validity of the reconstruction condition. Therefore the condition check needs to be done two times: once with the new edge's start vertex as s and once with its end vertex (see Figure 4.2).

If a new vertex is added to create a new triangle together with existing vertices v_1 and v_2, the new vertex has only these neighbors and does not need to be checked with the reconstruction condition – the umbrella consisting of three vertices can not have invalid connections. Nevertheless, the two new connections might have influence on the validity of the reconstruction condition for v_1 and v_2 as s, so these cases have to be checked (see Figure 4.3).

4.3.3 Deviation restriction

In contrast to the more obvious correlation between resampled edge and original polygon part in the two-dimensional case, in three dimensions this is not so trivial. The first approach was that the resampled triangle f is projected along its normal vector onto the

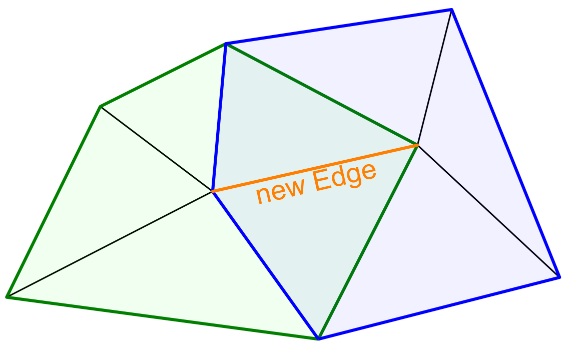

Figure 4.2: The two umbrellas that need to be checked with the reconstruction condition when an edge is added (if an umbrella is not complete yet, only the existing parts of it are tested)

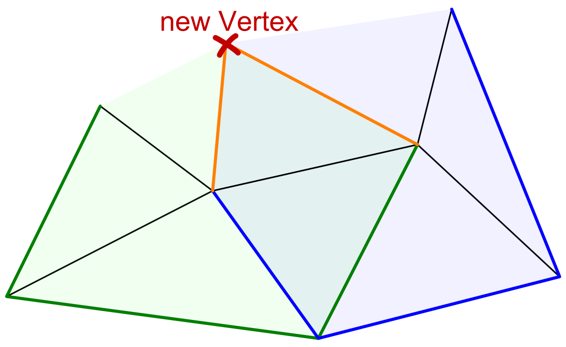

Figure 4.3: The two umbrellas that need to be checked with the reconstruction condition when a vertex is added (the umbrellas are not complete yet, only the existing parts of them are tested)

original mesh m, or in other words, the mesh part p_m correlated to a resampled triangle f is determined by a prism with f as its base (see Figure 4.4).

The problem that occurs with this approach is that some parts of the mesh are never regarded for deviation restriction (where the mesh is convex), and some parts are regarded multiple times (where the mesh is concave), as depicted in Figure 4.5. To counteract this problem, the sides of the prism were changed from *normal to the triangle* to *facing towards the middle of the angle between the two triangles adjacent to this resampled edge*. This assures that the correlated original mesh parts of two adjacent triangles connect seamlessly at resampled edges. So every resampled edge can be projected onto the original mesh dividing it into mesh parts that are correlated to the resampled triangles. Since the vertices are already placed on the original mesh, they do not need to be projected.

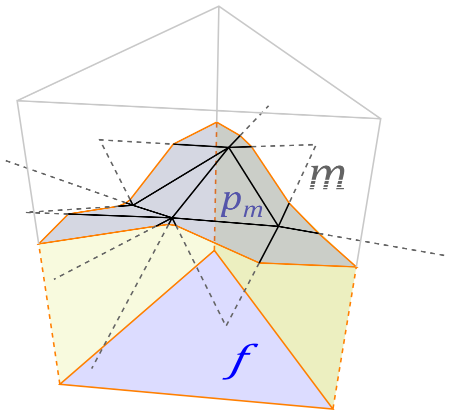

Figure 4.4: A resampled triangle projected onto the original mesh using a triangular prism

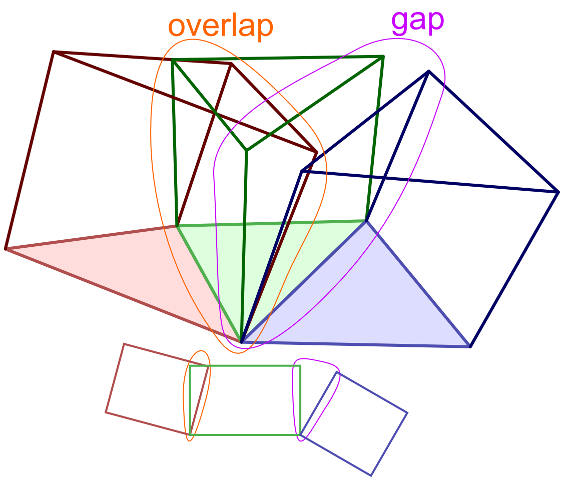

Figure 4.5: Three triangles of a resampled mesh (red, green and blue) and their corresponding prisms that determine the area of influence for the deviation computation. Below is an abstract illustration in side view that shows the overlap where the mesh is concave and the gap where the mesh is convex.

4.4 Implementation decisions for the 3-dimensional version

Contrary to the creation process of the resampled polygon in the algorithm's two-dimensional version, in this version it is not sufficient to create a new vertex in every iteration and connect it with the previously created vertex. The vertices also need to be connected with several other vertices to build a mesh. The key point for this algorithm was that in every iteration, a border edge of the already resampled mesh is regarded as the *current edge* e_{cur} (see Figure 4.6); for this edge, an area A on the original mesh that suffices the conditions and restrictions and would therefore be suitable for the new vertex is computed. The current edge e_{cur} has two other border edges n_1 and n_2 as neighbors. Their end points that are not part of e_{cur} are potential solutions s_1 and s_2 – either of those two vertices would require only one added edge e_{req} to build a new triangle together with e_{cur}. If s_1 or s_2 lie inside A the solution is valid and e_{req} can be added. It is also possible that another border vertex of the resampled mesh lies inside A; if this vertex is connected with e_{cur} the hole in the resampled mesh is split up into two holes. Otherwise a new vertex lying on the original mesh has to be created inside of A and is connected to e_{cur} with two edges.

To support the forming of a mesh with well formed triangles (that means as equilateral as possible), some formulas were developed that should be optimized with every newly created vertex. The new triangle that is formed with the new vertex is chosen with these formulas – triangles where all sides are of nearly equal length get higher scores. If the vertices are placed where their according triangles get the highest score, the resulting mesh's triangles will be more likely of equilateral form. These formulas might also be used upon the vertices s_1 and s_2 with a threshold to prevent that the connection is established even though that would lead to narrow, "deformed" triangles.

$$r = \frac{\textit{longest edge}}{\textit{shortest edge}}$$

$$r = \frac{abc}{8(s-a)(s-b)(s-c)}; s = \frac{a+b+c}{2}$$

Ideally, $r = 1$.

If $s_1 = s_2$, they both lie inside A and the resampled mesh is not only one triangle, then no edges have to be added any more to fill this hole in the resampled mesh; the triangle consisting of e_{cur} and s_1 is added and the algorithm has finished if there are no other holes left (i.e. there are no border edges to use as the next e_{cur}).

This approach has its problems; it does not consider which parts of the mesh are already resampled and therefore the resampled mesh might overlap at some points. If for example in Figure 4.6 A would contain s_2 so that this is connected instead of s_1, multiple resampled triangles would represent the same original mesh's part which is not desired.

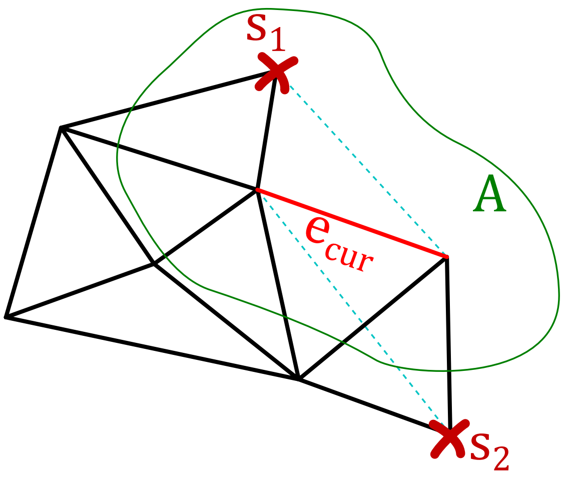

Figure 4.6: Creating an edge to one of the neighbor samples to get a new triangle; in the depicted case s_1 lies inside A and can therefore be used

4.4.1 Sharp corners and edges

In the three dimensional version, the homeomorphic condition prevents in some cases the construction beyond sharp edges, which therefore would need special treatment. Sharp corners can also occur when there are no sharp edges present (for example the top vertex of a cone could be a sharp corner, while the edges on its side are not sharp) and might also lead to problems, so these need to be checked separately. This thesis does not contain an approach specialized for the three-dimensional version.

4.4.2 Algorithm for the homeomorphic condition

As already mentioned in the chapters describing the algorithm's two-dimensional versions, the algorithm for the homeomorphic condition was designed to calculate the maximum fulfilling radius rather than just check if a certain radius applies.

The current sample, used as the center c of the homeomorphic sphere, usually lies somewhere on a triangle of the original mesh, rather than on an edge or even on a vertex – and also if this occasionally happens, it is just a special case and still can be viewed as lying on the triangle. Therefore it suffices to consider cases where c lies on a triangle.

For the triangle f_{start} that contains the center c the homeomorphic condition obviously applies: every point p_{fstart} on f_{start} can be reached from c with a path lying on the original mesh that has increasing distance from its start c (as path, the straight connection between c and p_{fstart} could be chosen – it lies on the triangle and straight line segments always have monotonically increasing distance from their start).

Every triangle has three neighbor triangles; the neighbor triangles of f_{start} are inserted into a list L_{hom} that stores triangles that still have to be checked. The edge that is shared with f_{start} is also stored as additional information; all the points on this edge fulfill the homeomorphic condition. This edge can be used as seed – if every point $p_{neighbor}$ on the neighbor triangle can be reached from the edge by a path with increasing distance to c then there also exist paths from c to every $p_{neighbor}$ with increasing distance.

Now, one of the triangles in L_{hom} is removed and tested, unless it has already been classified as fulfilling the homeomorphic condition. In the following description, the tested triangle is called f_{test} and its additionally stored edge (that fulfills the homeomorphic condition) is referred to by e_{test}. To simplify the visualization and explanation of the used method, the center c is projected onto the plane p_{test} in which the tested triangle f_{test} lies (the projected c is now called c_{proj}). This can be done because the distance of c from p_{test} would be used for distance computations inside the triangle, and since this distance does not change throughout the triangle it can be neglected by projection (see also Figure 4.7).

For the next few paragraphs, also see Figure 4.8.

All three edges of f_{test} (e_{test} and $e1_{test}$, $e2_{test} \neq e_{test}$) are used as separation borders into half-spaces. The three half-spaces that contain the triangle are further called h_e, h_{e1} and h_{e2} (named according to the used edge). If c_{proj} lies outside of h_e and inside of h_{e1} and h_{e2} the homeomorphic condition applies to the whole triangle. If c_{proj} lies outside of h_{e1} and h_{e2} (and therefore inside of h_e) the homeomorphic condition is not fulfilled for this triangle. The other cases need to be considered in more detail.

If c lies outside of h_{e1} it needs to be checked if every point on $e1_{test}$ can be reached with a path of continually increasing distance, starting at e_{test} respective the vertex where e_{test} and $e1_{test}$ meet (because this vertex is the nearest to c and is therefore the point that most likely fulfills this condition it suffices to test with this vertex). This results in a test if a line segment has continually increasing distance from a point – this sort of test has already been covered in the two-dimensional version of the algorithm. The $90\,^\circ$ check can be used again. If this test evaluates to true, then the homeomorphic condition is fulfilled for the whole triangle.

If c lies outside of h_{e2}, a similar test has to be performed using $e2_{test}$ instead of $e1_{test}$.

With these conditions, every case is covered and every triangle can be classified as fulfilling/not fulfilling the homeomorphic condition. If a tested triangle fulfills the condition, $e1_{test}$ and $e2_{test}$ and the according neighbor triangles are inserted into L_{hom}. The triangles in L_{hom} are tested until the list is empty.

Now the smallest distance from c to the triangles that were not classified as fulfilling the homeomorphic condition has to be computed (might be sped up using an appropriate data structure for the triangles – in the reference implementation an axis-aligned bounding box tree was used). The resulting distance is the largest valid radius.

4.4.3 Algorithm for the reconstruction condition

The validation of the reconstruction condition can be regarded as a set of inequations that has to be fulfilled. The conversion of the condition into a linear inequation system is shown in equation 4.1 on page 34. p is one of the current edges e_{cur} end points, q is a vertex of the umbrella around p that is not part of the new triangle and s is the new sample. s is required to lie on a triangle of the original mesh, so it was replaced by $v + a \cdot \vec{e_0} + b \cdot \vec{e_1}$ in the fourth line. This reduces the number of unknown variables

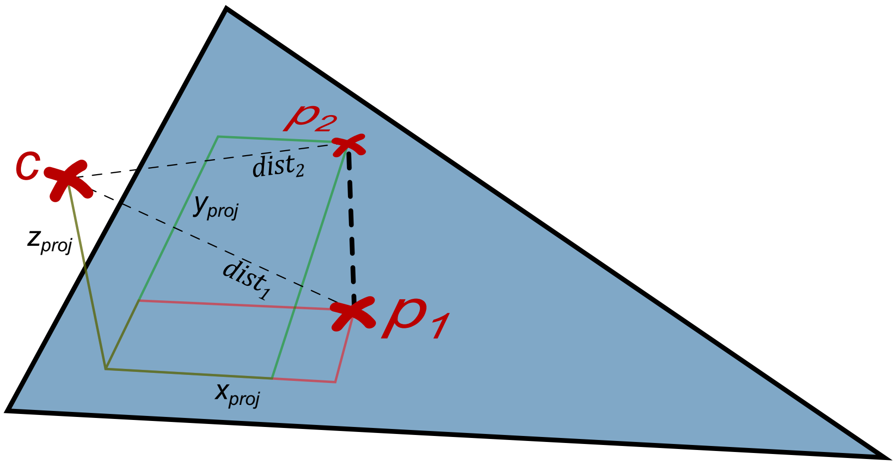

Figure 4.7: The projection of c onto the triangle's plane does not affect the distance computations needed to determine if a path lying on the triangle has increasing distance from c, because the omitted z-distance (z_{proj}) is the same for every point on the plane (like p_1 and p_2 in this figure)

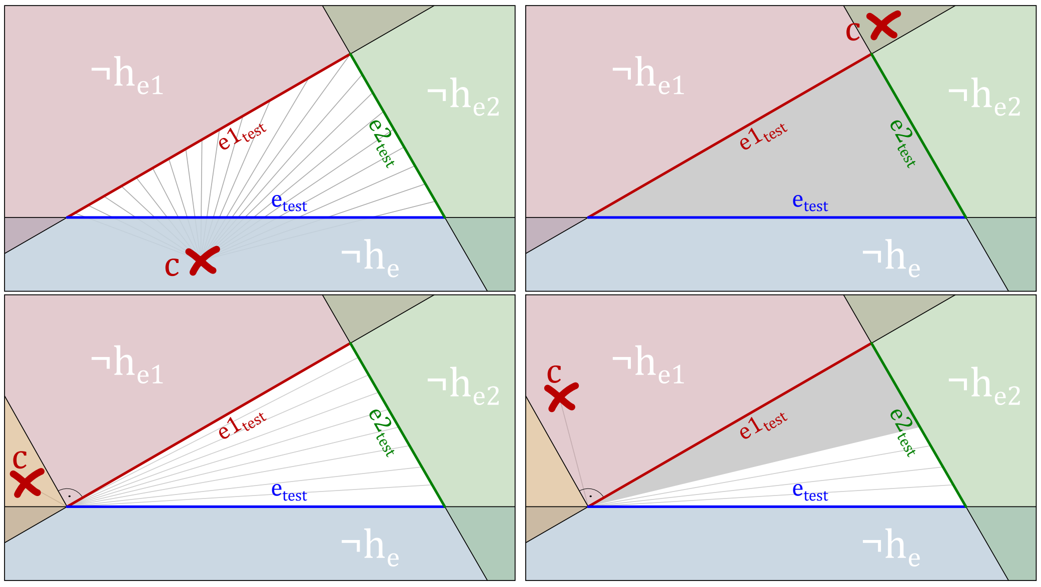

Figure 4.8: Different cases during homeomorphic condition check – the white area with gray rays can be reached by paths with increasing distance from c (the rays are example paths); the gray area is not reachable

from the three coordinates of s to the two variables a and b. If the restrictions $a + b \leq 1$, $a \geq 0$ and $b \geq 0$ are added the resulting linear programming problem only allows samples s that lie on the triangle with vertex v and spanning edges e_0 and e_1. To find the best solution it is necessary to create the linear programming problem for every original triangle that lies (partially) inside the homeomorphic radius and solve these problems using a rating function. One of the functions to support the construction of well-formed (nearly equilateral) triangles described in the beginning of section 4.4 on page 30 can be used for this task.

The representation of the task as a linear programming problem allowed the usage of already existing libraries that are dedicated to solve this sort of calculations efficiently. For the program written as part of this thesis, the open-source library NLopt was used [Joh], which also supports the solution of non-linear problems. It provides several optimization algorithms accessible with a unified interface, which allowed to experiment with the different algorithms. The fastest results for these tasks were achieved with the COBYLA-Algorithm [Pow94].

To add the distance restriction using the radius calculated with the homeomorphic condition algorithm some additional inequations have to be added as shown in equation 4.2. s is again the new sample, p is one of the two current edge's end points and r is the homeomorphic radius. The manhattan distance is used instead of the euclidean distance to attain linear equations. To implement the three absolute values without losing the functions continuity the inequation is split up into eight versions with all possible combinations of positive and negative values in the three absolute values. This has to be done for both p, therefore the homeomorphic condition adds 16 inequations to the linear programming problem.

$$\begin{aligned}
|p-s|^2 &< |q-s|^2 \\
p^2 - 2ps + s^2 &< q^2 - 2qs + s^2 \\
p^2 - q^2 &< 2s(p-q) \\
s = v + a \cdot \vec{e_0} + b \cdot \vec{e_1} \Rightarrow p^2 - q^2 &< 2(v + a \cdot \vec{e_0} + b \cdot \vec{e_1})(p-q) \\
p^2 - q^2 &< 2v(p-q) + 2a\vec{e_0}(p-q) + 2b\vec{e_1}(p-q) \\
\underbrace{p^2 - q^2 - 2v(p-q)}_{t} &< a\underbrace{(2\vec{e_0}(p-q))}_{u} + b\underbrace{(2\vec{e_1}(p-q))}_{v} \\
t &< a \cdot u + b \cdot v
\end{aligned} \tag{4.1}$$

$$\begin{aligned}
& & |s-p| &< r \\
s = v + a \cdot \vec{e_0} + b \cdot \vec{e_1} \Rightarrow & & |v + a \cdot \vec{e_0} + b \cdot \vec{e_1} - p| &< r \\
\text{manhattan distance} \Rightarrow & & |v_x + a \cdot \vec{e_0}_x + b \cdot \vec{e_1}_x - p_x| &+ \\
& & |v_y + a \cdot \vec{e_0}_y + b \cdot \vec{e_1}_y - p_y| &+ \\
& & |v_z + a \cdot \vec{e_0}_z + b \cdot \vec{e_1}_z - p_z| &< r
\end{aligned} \tag{4.2}$$

4.4.4 Algorithm for the deviation restriction

The deviation restriction could not be implemented by adding equations to the linear programming problem, because the shape of the resampled triangle has influence on the original vertices that have to be considered for the deviation computation. Depending on the shape, equations would have to be added or removed from the linear programming equation system. Simply multiplying the deviation by zero when the vertex does not have to be regarded would create discontinuity which leads to unsolvability of the equation system using linear programming. Because of these problems, the work on the deviation restriction was abandoned.

4.5 Conclusion

The results of the algorithm for two-dimensional cases are promising. The outputs contain, depending on the used parameters and the input polygon, less vertices than the original polygon, while approximating the given input with the desired deviation. The Bézier-curves can also be resampled with few vertices while maintaining the given maximum deviation. As a final reflection over the results of this thesis, in the following paragraphs some more figures of resampling results are shown and discussed. Also, the influence of the parameters will be compared. The parameters are described in section 2.2 on page 4.

In Figure 4.9 it can be seen that the resampling algorithm needs many samples to assure the edge retrieveability at sharp corners. If the *chamfering precision* is relaxed, the corners get cut off too early, leading to heavy information loss. If edge retrieveability is not needed, chamfering can be disabled; as a result, the sharp corners are preserved and no additional vertices are inserted that would only be needed for edge retrieveability.

As shown in Figure 4.10, without deviation restriction the input can be resampled with a very low amount of vertices – as long as it does not contain sharp corners with less than 60 °. The edge retrieveability is still maintained in the left image. Since the reference implementation of the algorithm does not support disabling the deviation restriction, the *maximum deviation* was set to a high number that will not be reached with this input.

In the last example (Figure 4.11), a case where the algorithm is able to reduce the vertex amount is shown. The vertices that approximately lie on a line can be replaced by only a few samples without losing important features. Even though the input contains a sharp corner below 60 °, the number of vertices could be reduced by over 37%.

This leads to the insight that the algorithm's ideal application field is polygons and Bézier-curves without or only with few sharp corners and if edge retrieveability is needed. The reduction of the vertex amount is only noticeable if the input polygon contains a significant amount of rather redundant vertices like in Figure 4.11.

While the two-dimensional versions of the algorithm are already applicable onto given polygons respectively paths, the step into three dimensions led to many additional problems that could not be solved satisfyingly in the scope of this thesis. The algorithm does not remember which parts of the original mesh are already resampled which can

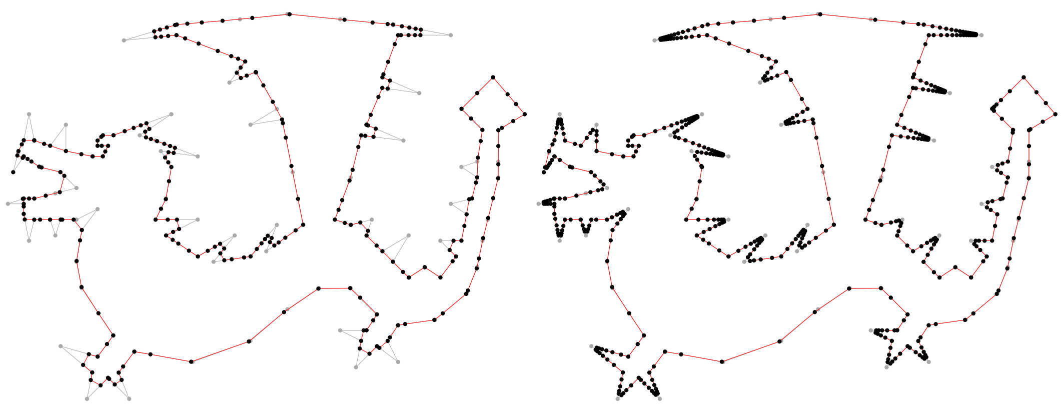

Figure 4.9: An example of a resampled polygon; on the right, the values of table 4.1 were used. The result contains 526 vertices. In the left image, *chamfering precision* was changed to 0.05. The result still contains 251 vertices, which is much more than the input with 130 vertices. This ratio is caused by the edge retrieveability and the already simplistic input.

cause overlaps, and the deviation restriction could not be adequately transferred into the three dimensional version.

For further work on the three-dimensional algorithm it might be recommendable to find a deviation definition that can be directly depicted as a set of inequations in the linear programming problem.

The code is available on the following svn-repository:

`https://projects.cg.tuwien.ac.at/data/svn/svnohrhallinger/sampling2d`

`https://projects.cg.tuwien.ac.at/data/svn/svnohrhallinger/sampling3d`

For only the `.exe` that is executable on windows, use the following svn-url:

`https://projects.cg.tuwien.ac.at/data/svn/svnohrhallinger/sampling2d/Application`

chamfering	true
use reconstr. cond	true
chamfering precision	0.01
discretion precision	0.0001
maximum deviation	0.01

Table 4.1: Algorithm settings

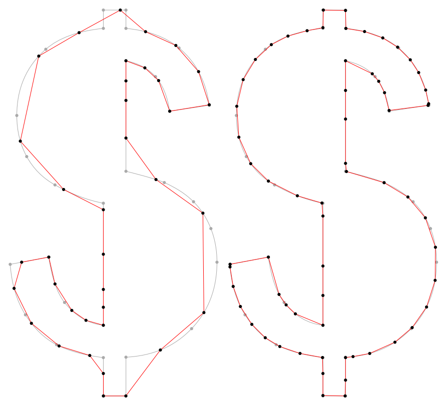
Figure 4.10: An example of a resampled sequence of Bézier-curves; on the right, the values of table 4.1 were used. The result contains 64 vertices. In the left image, *maximum deviation* was changed to 100.01, which reduces the number of vertices to 37.

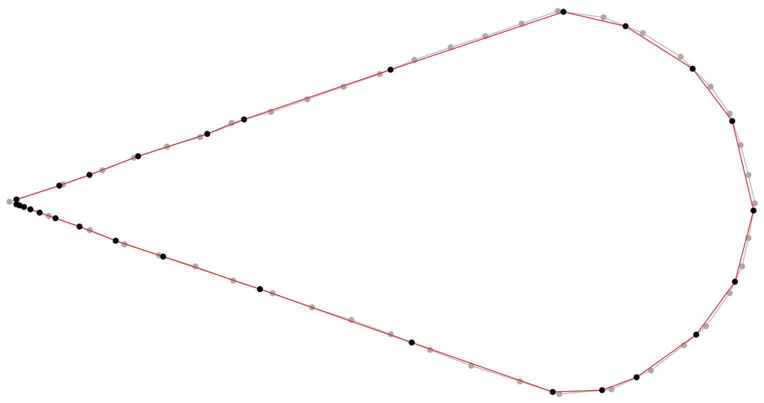
Figure 4.11: This time, the input polygon contains some rather redundant vertices. While again using the same values as in table 4.1, the number of vertices was reduced from 45 to 28 with the algorithm – even though the polygon contains a sharp corner below 60 °.

Bibliography

[BF84] Richard L. Burden and J. Douglas Faires. 2.1 the bisection algorithm. *Numerical Analysis*, 1984.

[BF85] Richard L. Burden and J. Douglas Faires. 2.1 the bisection algorithm. *Numerical Analysis*, 1985.

[Hop96] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM, 1996.

[Joh] Steven G. Johnson. The nlopt nonlinear–optimization package. `http://ab-initio.mit.edu/nlopt`. Accessed: 2014-09-12.

[OM13] Stefan Ohrhallinger and Sudhir Mudur. An efficient algorithm for determining an aesthetic shape connecting unorganized 2d points. *Computer Graphics Forum*, 32(8):72–88, 2013.

[otINRCI] Visual Computing Lab of the Italian National Research Council (ISTI). Visualization and computer graphics library (vcg). `http://vcg.sf.net`. Accessed: 2014-09-14.

[Pow94] Michael James David Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer, 1994.

[RB93] Jarek Rossignac and Paul Borrel. Multi-resolution 3d approximations for rendering complex scenes. In Bianca Falcidieno and TosiyasuL. Kunii, editors, *Modeling in Computer Graphics*, IFIP Series on Computer Graphics, pages 455–465. Springer Berlin Heidelberg, 1993.