Methoden der Diskreten Mathematik bei der Rekonstruktion der Topologie eines CAD-Datenmodells

Diplomarbeit
von
Thomas Willhalm

vorgelegt an der
Universität Konstanz
Fakulät für Mathematik und Informatik

1998

Contents

1 Introduction **2**

2 Problem Description **6**
2.1 Input 6
2.2 Output 9

3 Background **12**
3.1 Need for topology 12
3.2 Known Approaches 13
3.3 Ill-Posedness 14

4 Reconstruction Using Error Estimation **16**
4.1 Methodology 16
4.2 Computational Results 20

5 The Algorithm **24**
5.1 Description 24
5.1.1 Stage 1: Neighboured Mesh Elements 25
5.1.2 Stage 2: Neighboured Edges 26
5.1.3 Stage 3: Conflicts between Neighbours I 26
5.1.4 Stage 4: Conservative Error Corrections 28
5.1.5 Stage 5: Conflicts between Neighbours II 30
5.2 Accuracy 31
5.3 Complexity 36
5.4 Computational Result 37
5.5 Parameter Intervals Revisited 40

6 Conclusion **41**

A Diagrams about Relative Error Estimation **43**

B Images of Workpieces **47**

Chapter 1

Introduction

Computer-aided design plays an important role in today's engineering. In this thesis, we deal with CAD data models such as the one shown in Figure 1.1. Each model consists of *mesh elements* and approximates the surface of a workpiece. In general, the mesh elements are not parts of a plane, and their edges are not straight lines. More precisely, *trimmed parametric surface patches* were used in the data available to us.

One of the tasks which is to be done automatically is the reconstruction of the so-called *topology* of the CAD data model, i.e. the information whether and where two mesh elements are to be regarded as immediately neighboured. Many wide-spread data formats for CAD models do not provide the neighbourhoods. The topology of a CAD model is important, since almost every further step of the CAD process relies on this information. Section 3.1 gives examples of such steps.

Pictures such as Figure 1.1 or the images in Appendix B suggest that the edges of neighboured mesh elements fall together geometrically. This is generally not the case. There are normally gaps between the mesh elements – gaps that can be as large as in Figures 1.2 and 1.3. These gaps are sometimes even larger than mesh elements in the same workpiece. They are the reason why the problem is non-trivial, since we have no knowledge whether an edge is neighboured to one, several or no other edges (Figure 2.3). Consider two edges that are situated next to each other. The gap between them may have been intended by the engineer who designed the workpiece, but the two surface patches may also be regarded as parts of one closed surface. As a consequence, the quality of the output is the point of interest. In fact, time and space consumption is negligible compared to certain other steps of the CAD process.

In Chapter 2, we will describe the input and the output of this problem in detail. Although the topic of the thesis touches several disciplines (discrete and analytical computational geometry, numerical analysis, engineering), the core of the thesis belongs to the field of discrete computational geometry. For clarity of exposition, we will essentially ignore the technical problems caused by numerical calculations and formulate the algorithm as if infinite precision were available and geometric constructions such as the perpendicular onto a curve were always well defined and unique.

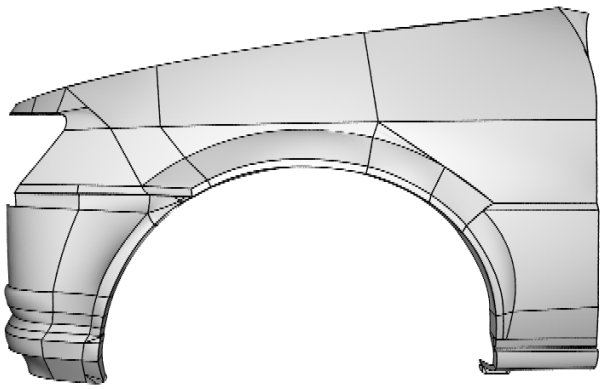

Figure 1.1: This mudguard (workpiece 8) is a typical example of our real world instances. It consists of surfaces patches (the so-called mesh elements). The black curves indicate the edges of the mesh elements.

In Chapter 4 of this thesis, we focus on a common approach, which is highly suggested by the images of the workpieces (see Appendix B): to define a distance measure of two edges and choose a threshold value. An example of a distance measure is the maximal distance between two points on the respective edges, or the sum of the distances between the end points. Only pairs of edges whose distance is smaller than the threshold value are considered as neighboured. This may be viewed as estimating the *absolute error* of the placement of the edges.

To refine this approach, one can replace the absolute error with the *relative error*. Properties such as the lengths of the edges, or the area or perimeter of the mesh elements seem reasonable scalings for such a scaled distance measure. To our knowledge, these approaches with scaled or unscaled distance measures are the main strategy in today's industrial practice [7, 8]. Even Figures 1.2 and 1.3 suggest that this approach yields a very good approximation of the right neighbourhoods. Thus the problem does not seem to deserve further research.

The main contribution of Chapter 4 is to demonstrate empirically that in general this approach fails against all expectations – even if we assume that the algorithm automatically finds the individually best threshold value for each edge (of course, this assumption is far too optimistic). We have examined a variety of unscaled distance measures (i.e. absolute error estimation) and possible extensions to scaled distance measures (i.e. relative error estimation). Naturally, our investigations only cover a subset of all possible distance measures. However, we believe that the most logical ones were treated and that they are good representatives for all reasonable distance measures, because our results are not too sensitive to the choice of the distance measure. This will be discussed further in Section 4.1.

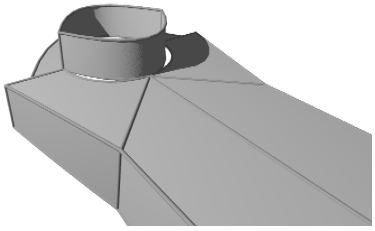

Figure 1.2: An example of a sloppy design (workpiece 15). The complete workpiece is shown in Figure B.15

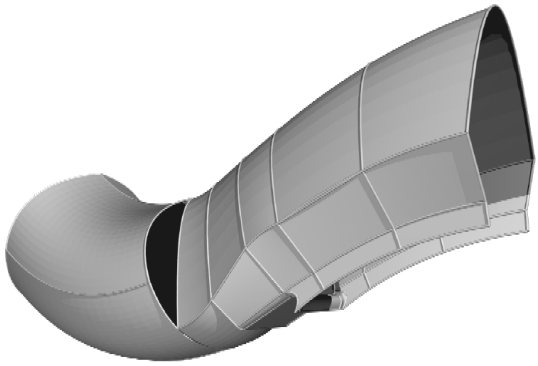

Figure 1.3: An extreme gap from the interior of Figure 3.1. The black, semicircular hole does not make any sense from an engineering point of view and might be the result of a format conversion.

Our results in Chapter 4 have been presented at WAE'98 [11]. They imply that an algorithm that is based solely on some measure for the (relative or absolute) error cannot determine a satisfying approximation of the topology. In other words, unstructured information such as the position, size, and shape of mesh elements does not suffice to decide whether or not two mesh elements are neighboured. One seems forced to look at the problem from a more abstract point of view and use *discrete techniques*. By that we mean that rules are applied in which geometrical and numerical details are abstracted away: some kind of *logical inference rules*, which incorporate more than two mesh elements in their reasoning. A simple example is the rule that the relation of neighbourhoods has to be transitive: if patches A and B as well as patches B and C are neighboured, then A and C must also be neighboured. Such a rule might detect missing neighbourhoods along branchings like in Figure 2.3. Furthermore, an algorithm may use meta-information about the structure, the topology, of the workpiece. The special case where the CAD model is the surface of a (connected) solid object or a plane deformed by a press is an example of strong meta-information: apart from sophisticated cases, every edge has exactly one neighbour. Hence in this case, the problem can be reduced to a weighted matching problem. However, we will treat the general case and therefore cannot use this model.

In Chapter 5, an algorithm is presented for the reconstruction of the topology, which relies on structural techniques and which produces acceptable results for our benchmarks. Furthermore, this approach allows an easy check of the output. By this, it is possible for the engineer to find and correct the few remaining errors in the output. The quality of the algorithm – in terms of output quality and time consumption – and the correctness of the checker will be substantiated by theorems. Furthermore benchmark results will be given for a prototyped implementation.

The algorithm consists of several stages. Each stage tries to ensure that the result will be an *overestimation* of the correct result, i.e. that the algorithm provides at least the correct neighbourhoods (and possibly some wrong ones). The first stage finds a set of candidates for neighboured mesh elements. This is done by projecting the mesh elements on lines and examining the intersections of these projections. In the following stages, a set of neighboured edges is found and stage per stage reduced to the final output. Apart from a classical error estimation, they include logical inference rules to remove pairs of edges. They use the fact that two edges of the same mesh element cannot be neighboured to the same part of another mesh element, and they make sure that the generated neighbourhoods are a transitive relation. Both observations are suited to determine pairs of neighbours in the set that are not neighboured and remove them from the list.

For each stage sufficient conditions will be provided that guarantee an overestimation of the neighbourhoods. This request is sufficient for the provided checker to work correctly and is desirable for other tasks that use the generated topology. The visualization by the checker is also necessary, because the decision whether two edges are neighboured can be subjective in some cases (Figure 1.3). The algorithm and the checker were presented at ESA'97 [10].

In summary, we were faced with a practical problem that does not seem to be mathematical in nature: the dirtiness of the problem cannot be abstracted away, since it is at the heart of the problem. Errors that caused the common approach to fail were found all over the test set. Nevertheless, it is possible to use mathematical techniques to get acceptable results and check them efficiently.

Chapter 2

Problem Description

2.1 Input

We consider surface models like for example hollow bodies that consist of metal plates (See Appendix B to get an impression of the workpieces.) The surface of a workpiece is a set M of surfaces patches. Each surface patch $m \in M$ is a bordered regular two-dimensional manifold in the three-dimensional Euclidean space $\mathbb{R}^3$. The border of a mesh element m is partitioned into differentiable curves $e \subset m$ – the edges of m. The mesh elements M together with their edges E form one input instance.

To give a concrete, illustrative example, the format of the data that is available to us is described. A CAD model consists of a set of trimmed parametric surface patches. In our case, a patch can be three or four sided (Figure 2.1).

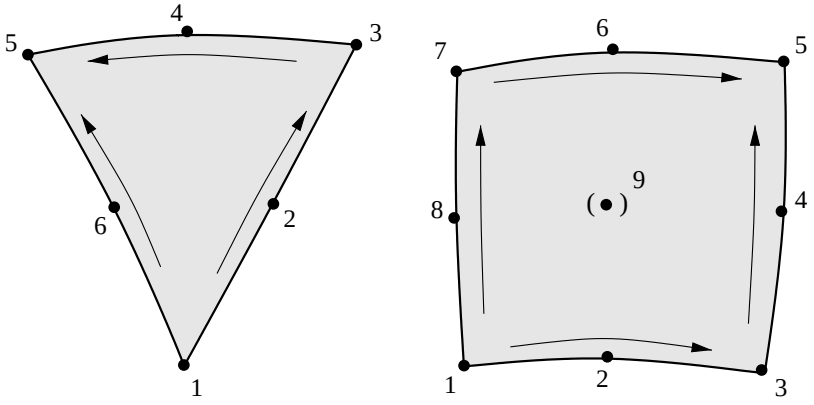

Figure 2.1: a triangular mesh element with six supporting points and a quadrilateral mesh element with eight supporting points on the boundary and one (optional) point in the interior. The arrows indicate the direction of the parameterization of the edges.

More precisely, the shape of a surface patch is defined via a function

$$\begin{aligned} \mathcal{D} &\to \mathbb{R}^3 \\ (u,v) &\mapsto \sum_{i=1}^{n} w_i(u,v)\, p_i. \end{aligned}$$

The vectors $p_i \in \mathbb{R}^3$ are the points on the boundary according to Figure 2.1. In case of a three sided patch, the domain of definition is $\mathcal{D} := \{(u,v) \in [0;1]^2 \ ; \ u+v \leq \frac{1}{2}\}$ and the weights are:

$$\begin{aligned} w_1^{(3)}(u,v) &= (1-u-v)(1-2u-2v) & \qquad w_4^{(3)}(u,v) &= 4uv \\ w_2^{(3)}(u,v) &= 4u(1-u-v) & \qquad w_5^{(3)}(u,v) &= v(2v-1) \\ w_3^{(3)}(u,v) &= u(2u-1) & \qquad w_6^{(3)}(u,v) &= 4v(1-u-v) \end{aligned}$$

For a four sided mesh element, the domain of definition is $[-1;1]^2$ and the weights are:

$$\begin{aligned} w_1^{(4)}(u,v) &= \tfrac{1}{4}(u^2-u)(v^2-v) & \qquad w_6^{(4)}(u,v) &= -\tfrac{1}{2}(u^2-1)(v^2+v) \\ w_2^{(4)}(u,v) &= -\tfrac{1}{2}(u^2-1)(v^2-v) & \qquad w_7^{(4)}(u,v) &= \tfrac{1}{4}(u^2-u)(v^2+v) \\ w_3^{(4)}(u,v) &= \tfrac{1}{4}(u^2+u)(v^2-v) & \qquad w_8^{(4)}(u,v) &= -\tfrac{1}{2}(u^2-u)(v^2-1) \\ w_4^{(4)}(u,v) &= -\tfrac{1}{2}(u^2+u)(v^2-1) & \qquad w_9^{(4)}(u,v) &= (u^2-1)(v^2-1) \\ w_5^{(4)}(u,v) &= \tfrac{1}{4}(u^2+u)(v^2+v) \end{aligned}$$

If the ninth point in a four sided surface patch is omitted, it is replaced by the mean value of the eight other points. One easily verifies that for all three types of mesh elements the sides of a surface patch – its trimming curves – are parabolic curves. Up to a parameter transformation every edge e can be represented by

$$\begin{aligned} f_e : \ [-1;1] &\to \mathbb{R}^3 \\ t &\mapsto \sum_{i=1}^{3} w_i^{(1)}(t)\, p_i, \end{aligned}$$

where

$$\begin{aligned} w_1^{(1)} &= \frac{1}{2}(t^2-t) \\ w_2^{(1)} &= 1-t^2 \\ w_3^{(1)} &= \frac{1}{2}(t^2+t) \end{aligned}$$

The parameter of an edge is therefore in the interval $[-1;1]$. The *orientation* of an edge is given by the numbering of the defining points according to Figure 2.1.

A four sided surface patch is a special case of the well known bicubic Beziér patch [9], which is defined by the function:

$$\begin{array}{rcl} [0;1]^2 & \rightarrow & \mathbb{R}^3 \\ (u,v) & \mapsto & UMBM^tV^t \end{array}$$

with $U = (u^3 \quad u^2 \quad u \quad 1)$, $V = (v^3 \quad v^2 \quad v \quad 1)$ and

$$M = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

The parameters in the matrix B are elements of $\mathbb{R}^3$. The four sided mesh elements can be represented in an equivalent way:

$$\begin{array}{rcl} [0;1]^2 & \rightarrow & \mathbb{R}^3 \\ (u,v) & \mapsto & UNPN^tV^t \end{array}$$

where

$$N = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -2 & -2 & 4 \\ 0 & 3 & 1 & -4 \\ 0 & -1 & 0 & 0 \end{pmatrix} \qquad \text{and} \qquad P = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & p_1 & p_7 & p_8 \\ 0 & p_3 & p_5 & p_4 \\ 0 & p_2 & p_6 & p_9 \end{pmatrix}$$

This representation incorporates already a change of the domain of definition. Now, the transformation from a four sided mesh elements to a bicubic Beziér patch can be calculated by two matrix multiplications:

$$B = \underbrace{M^{-1}N}_{D} PN^t(M^t)^{-1} = DPD^t$$

with

$$D = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & -\frac{4}{3} \\ 0 & \frac{1}{3} & 0 & -\frac{4}{3} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

We obtain:

$$B = \begin{pmatrix} p_1 & -\frac{1}{3}p_7 + \frac{4}{3}p_8 & -\frac{1}{3}p_1 + \frac{4}{3}p_8 & p_7 \\ -\frac{1}{3}p_3 + \frac{4}{3}p_2 & \frac{1}{9}p_5 - \frac{4}{9}p_6 - \frac{4}{9}p_4 + \frac{16}{9}p_9 & \frac{1}{9}p_3 - \frac{4}{9}p_2 - \frac{4}{9}p_4 + \frac{16}{9}p_9 & -\frac{1}{3}p_5 + \frac{4}{3}p_6 \\ -\frac{1}{3}p_1 + \frac{4}{3}p_2 & \frac{1}{9}p_7 - \frac{4}{9}p_6 - \frac{4}{9}p_8 + \frac{16}{9}p_9 & \frac{1}{9}p_1 - \frac{4}{9}p_2 - \frac{4}{9}p_8 + \frac{16}{9}p_9 & -\frac{1}{3}p_7 + \frac{4}{3}p_6 \\ p_3 & -\frac{1}{3}p_5 + \frac{4}{3}p_4 & -\frac{1}{3}p_3 + \frac{4}{3}p_4 & p_5 \end{pmatrix}$$

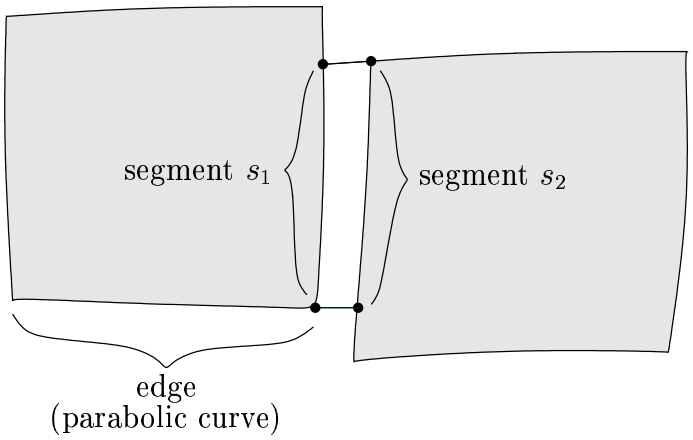

Figure 2.2: Two neighboured mesh elements and the segments, s_1 and s_2, that constitute this neighbourhood.

2.2 Output

In this section, a formal definition of a feasible output is given. Some notions, which will reappear in the description of the algorithm, are introduced as well.

Definition 1 (neighbourhood set)
Recall that E is the set of edges in the input. We denote the power set of $[-1;1]$ by $\mathcal{P}([-1;1])$. A neighbourhood set of the workpiece is then defined as a triple (N, D, O) where

$$\begin{array}{rcl} N & \subset & E \times E \\ D : N & \to & \mathcal{P}([-1;1]) \times \mathcal{P}([-1;1]) \\ O : N & \to & \{true, false\}. \end{array}$$

If $(e_1, e_2) \in N$ we call e_1 and e_2 neighboured. The triple $((e_1, e_2), D(e_1, e_2), O(e_1, e_2))$ is sometimes called a neighbourhood.

The boolean function O chooses whether or not two neighboured edges are orientated in the same direction. Figure 2.2 demonstrates that edges need not to be neighboured on a whole edge. For neighbours (e_1, e_2), the parts of the edges where they are neighboured will be given by $D(e_1, e_2)$ using the parameterization of the edge that has been introduced in the previous section. Remember that the edge e_1 is given by a function

$$\begin{array}{rcl} f_{e_1} : \ [-1;1] & \to & \mathbb{R}^3 \\ t & \mapsto & \sum_{i=1}^{3} w_i^{(1)}(t)\ p_i. \end{array}$$

The function D provides the subsets of the parameter intervals where two edges are neighboured. The part of e_1 that constitutes to the neighbourhood is then given by restricting the domain of definition of the function f_{e_1} to the first component of $D(e_1, e_2)$. Analogously, the part of e_2 is given by the second component of $D(e_1, e_2)$. We will denote the respective components by $D_1(e_1, e_2)$ and $D_2(e_1, e_2)$. However, we make the following assumption:

Assumption 2
For every pair (e_1, e_2), every component of $D(e_1, e_2)$ is connected, closed and contains more than one point.

Such a connected subset of an edge will be called a *segment*. Consider a segment of an edge e and the interval $[a; b] \subseteq [-1; 1]$ that defines the segment. The points $f_e(a)$ and $f_e(b)$ in the segment are called the *end points* of the segment.

Obviously, a neighbourhood set (N, D, O) only makes sense, if it is symmetric, i.e. if $(e_1, e_2) \in N$ then $(e_2, e_1) \in N$ with $D_1(e_1, e_2) = D_2(e_2, e_1)$, $D_2(e_1, e_2) = D_1(e_2, e_1)$, and $O(e_1, e_2) = O(e_2, e_1)$. So, adding (e_1, e_2) to a neighbourhood set implies adding (e_2, e_1) and removing (e_1, e_2) implies removing (e_2, e_1) without further notice. Furthermore, we implicitly assume N to be reflexive with $D(e, e) = ([-1, 1], [-1, 1])$ and $O(e, e) = \text{true}$ for all edges $e \in E$.

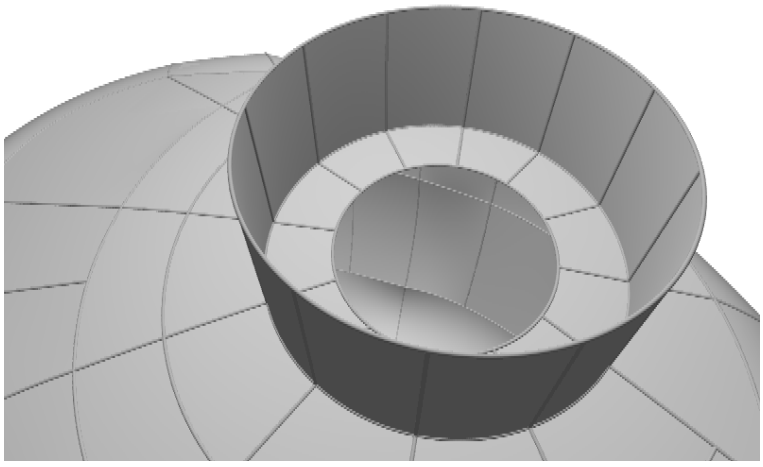

Figure 2.3: A mesh element can have edges that are not neighboured to any other edge. They may be around a *hole* or along a *rim*. Edges can also have more than one neighbour along so-called *branchings* as depicted in this figure.

Definition 3 (feasible output)
A neighbourhood set (N, D, O) that fulfills Assumption 2 with N being an equivalence relation is called a feasible output*.*

Assumption 2 implies that for every pair of edges there are at most two neighboured segments. On the other hand, not only two, but three or even more mesh elements may have an edge in common (Figure 2.3). Figure 2.3 also shows a rim and a hole as two exemplary configurations where a mesh element is not neighboured to any other mesh element on an edge.

We can now define the semantics of (N, D, O). For that, remember that $f_{e_i} : [-1, 1] \to \mathbb{R}^3$ is the function that determines the shape of e_i as defined in Section 2.1.

Definition 4 (corresponding points and degree)
Let (N, D, O) be a feasible output, $(e_1, e_2) \in N$, and $([t_1, h_1], [t_2, h_2]) = D(e_1, e_2)$. If for $x_1 \in [t_1, h_1]$

$$x_2 := \frac{x_1 - t_1}{h_1 - t_1}(h_2 - t_2) + t_2 \quad \text{if } O(e_1, e_2)$$
$$x_2 := \frac{h_1 - x_1}{h_1 - t_1}(h_2 - t_2) + t_2 \quad \text{if not } O(e_1, e_2)$$

then we say that the point $f_{e_1}(x_1) \in e_1$ corresponds to *$f_{e_2}(x_2) \in e_2$* according to *(N, D, O), and vice versa. The number of the corresponding points for p $+1$ will be called the* degree *$\deg_{(N,D,O)}(p)$.*

For example, in Figure 2.3 points with different degrees are shown. The points along the rim and around the whole have degree 1, the points along the branching have degree 3, and the points of all other edges have degree 2. A colouring according to the degrees of the points on the borders will provide a possibility to check the result of the output is certain situations. This will be discussed in Section 5.2.

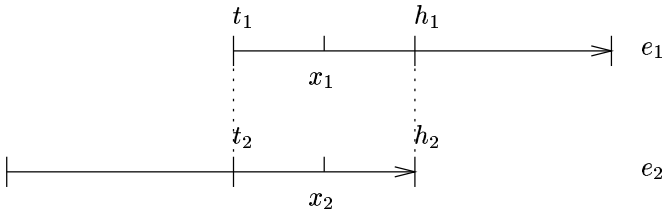

Figure 2.4: A point $x_1 \in e_1$ corresponds to a point $x_2 \in e_2$ (according to a feasible output (N, D, O)) if $(x_1, x_2) \in D(e_1, e_2)$. An exact description is given by Definition 4.

Chapter 3

Background

3.1 Need for topology

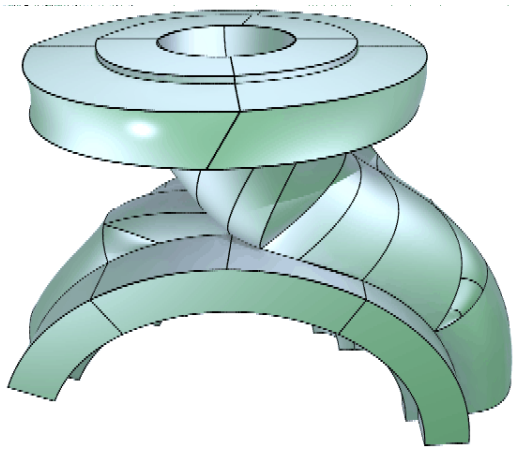

Figure 3.1: A CAD model of a pump (workpiece 13).

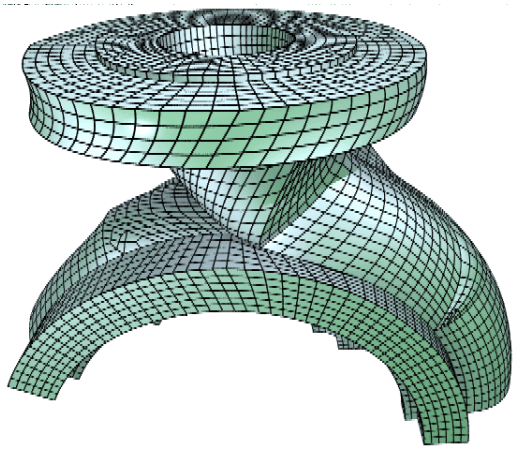

Figure 3.2: The same workpiece refined by the algorithm in [5].

The topology of a CAD model is essential for most automated tasks:

- For example, applying finite element methods only makes sense when forces or heat flow are transmitted correctly from element to element.
- The topology is also necessary to refine the mesh of a workpiece. Figure 3.1 shows an example of a CAD data model that was constructed manually, and a refined mesh of the same workpiece that was automatically produced by the algorithm in [5].
- On the other hand, it is sometimes desirable to coarse a mesh. This may be necessary to reduce the amount of data for scanned workpieces. Another application is to

replace mesh elements by others of a simpler type (for example, to substitute a set of triangles that approximate a sphere by a sphere).

- Furthermore, the topology is needed for smoothing surfaces. This is done by adjusting the normal vectors of neighboured mesh elements.

See [7] for a list of further tasks for CAD data models that describe the surface of a solid object.

3.2 Known Approaches

Existing approaches. Some restrictions on the input format as well as much cleaner CAD models sometimes allow the use of a distance measure in combination with a threshold value. Sheng and Meier examined in [8] the case when the surface is the boundary of a solid object. They also faced the problem of gaps that are as large as mesh elements. Their tool delegates cases of doubt to the user.

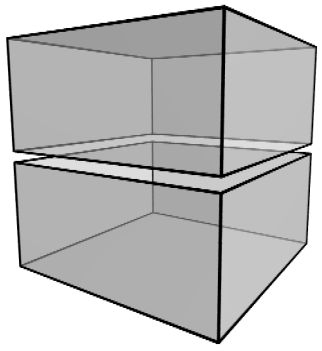

Figure 3.3: The gaps between the upper and the lower part cannot be found be the algorithm of Bøhn and Wozny, because the parts are not connected.

Knowledge about the discrete structure is used in very few existing approaches. In [2] Bøhn and Wozny restrict the problem to workpieces where the topology is already known for big parts. In practice, this is of course less difficult than a total reconstruction of the topology. Apart from this, their approach is heavily based on the assumption that the mesh elements form the surface of a solid object. It incorporates the fact that the closed surface is orientable and their approach uses the information on which side of the surface the solid object is situated. The problem can therefore be reduced to a matching problem. However, their algorithm can only find a special kind of gaps: it cannot cope with gaps between objects that belong to different connected components (See Figure 3.3).

The same problem was also treated by Barequet and Sharir in [1]. Their approach is not limited to a special kind of gaps. The candidates for neighbourhoods are found by a threshold value. They then try to choose neighbourhoods that permit to correctly orientate the surface.

3.3 Ill-Posedness

Gaps. The main problem when finding neighbourhoods is that usually the CAD models contain significant gaps. These gaps have to be distinguished from holes that are intended by the designer of the CAD model: consider two edges that are situated close to each other. The algorithm has to decide whether they are intended to be neighboured or just two separate edges with a hole in between. Since the gaps can be as large as in Figure 1.2 and Figure 1.3, this decision becomes a serious problem.

The problem is made even more difficult by the occurrence of branchings (Figure 2.3). Consider an edge e. We already have found a neighbour e' of this edge. If we want to decide whether it is neighboured to a third edge e'', the knowledge of the neighbourhood (e, e') does not help our decision. The pair (e, e'') can nevertheless be a correct neighbourhood.

Reasons for gaps. The gaps in Figure 1.2 and Figure 1.3 are obviously not due to mere numerical rounding errors. We have found several explanations why these and other gaps exist:

- While creating a workpiece, the designer can position the edges freely in the three dimensional space: as depicted in Figure 2.2, two edges can be neighboured on parts of their full length. The low order of the polygons that model the edge makes gaps unavoidable then.

- Conversions between non-isomorphic data formats sometimes make the approximation of a free form surface by another type necessary. For example, an approximation of a circle by a parabola may also result in gaps like in Figure 1.3.

- Automatic tools that create CAD models from input devices like computer tomographs or 3d scanners do not have any knowledge about the topology of the workpiece. Errors and an inaccurate precision of measurement can then lead to gaps.

See [3] for a detailed discussion of these issues.

Discrete information. We have found gaps that are larger than mesh elements of the same workpiece. This has some unpleasant consequences, which are depicted in Figure 3.4: on the left side, the upper and lower mesh elements are to be regarded as neighboured, whereas this is not the case on the right side, because of the triangle between them. An algorithm cannot decide whether the mesh elements are neighboured, if it does not have any information about the triangle. As a consequence, the properties of the two considered mesh elements are not sufficient for an algorithm to take its decision.

Not only a mesh element between an examined pair of edges, but also the surrounding mesh elements must be incorporated into the interpretation of a local configuration. For

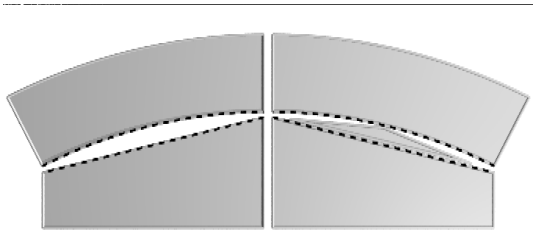

Figure 3.4: On the left side, the mesh elements are neighboured, but their mirror-symmetric counterparts on the right side are not, due to the triangle between them.

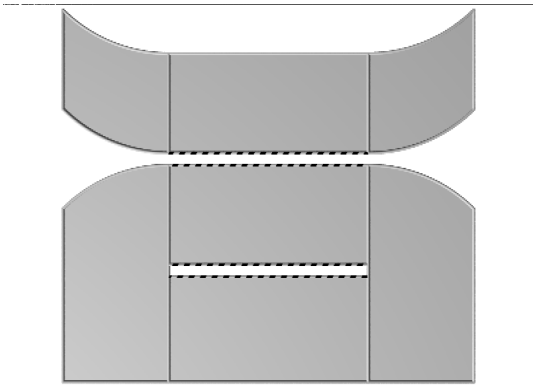

Figure 3.5: Two gaps of the same width and length. The surrounding mesh elements are a hint that the upper one is probably intended whereas the lower one is an error.

example, in Figure 3.5 two gaps of the same length and width are shown. However, in the upper case the gap continues on both sides and in the lower case the gap closes. This is a strong argument that the first gap is intended as opposed to the second one. Roughly speaking, all of these considerations are examples of what we mean by incorporating discrete information.

Claim. We have seen that in certain situations discrete information must be used. Based on a systematic computational study, we will show in the next Chapter that such situations occur too often to be ignored: they are typical, not pathological. In Chapter 5 an algorithm is presented that incorporates discrete information. In particular, the situation depicted in Figure 3.4 is handled correctly by our algorithm.

Chapter 4

Reconstruction Using Error Estimation

As shown in Section 3.3 there exist significant gaps between the mesh elements of instances. This chapter presents the surprising result that the gaps in real world data prevent the success of the approach that suggests itself: to define a more or less complicated distance function of edges and distinguish non-neighboured pairs of edges from neighboured pairs by a threshold value. Our investigations suggest that this approach does not provide satisfying results for data that is as "dirty" as the workpieces available to us.

4.1 Methodology

Test cases. The workpieces that have been examined stem from industrial applications. Unfortunately it is quite difficult to get such instances because industrial companies keep them confidential. However, the number of test cases is still high enough to demonstrate the trend to a negative result. We also know that the test cases were created using diverse CAD packages in various companies. Thus, we conclude that the examined effects are not tailored to a specific construction tool or engineer.

In our opinion, it is impossible to systematically generate artificial instances that model realistic workpieces: a random set of parametric surface patches might never resemble a workpiece like a pump or a console. If we introduced artificial gaps into closed surfaces to imitate the dirtiness of the data, we would not have examined the nature of common errors, but our interpretation of them.

Examined class of algorithms. For the computational study, we consider the class of algorithms that work according to the following pattern: an algorithm decides whether or not a pair of edges is neighboured by comparing the distance of the edges with a threshold value. The algorithms differ only in the used distance measure. Furthermore, we assume that the algorithms always find the best threshold value for each workpiece. In case of scaled distance functions, we even assume the choice of the best threshold value for each single edge. Both assumptions are by far more than one can expect from an algorithm without human interaction.

Reference neighbours. We use a set of *reference neighbours* for making statistics and assessing the output of different algorithms. Since the problem is so ill-posed, it is impossible to automatically check the output of an algorithm otherwise. The reference neighbourhoods were produced by a commercial CAD tool [4] and the algorithm that is presented in Chapter 5. The result was checked manually (using Theorem 9). Where necessary the neighbourhoods were corrected by hand in this project and the project of [5]. In fact, we can assume, that the reference neighbours of a workpiece are identical with the correct neighbours $(\mathcal{N}, \mathcal{D}, \mathcal{O})$ – the ones that the designer of the workpiece had in mind.

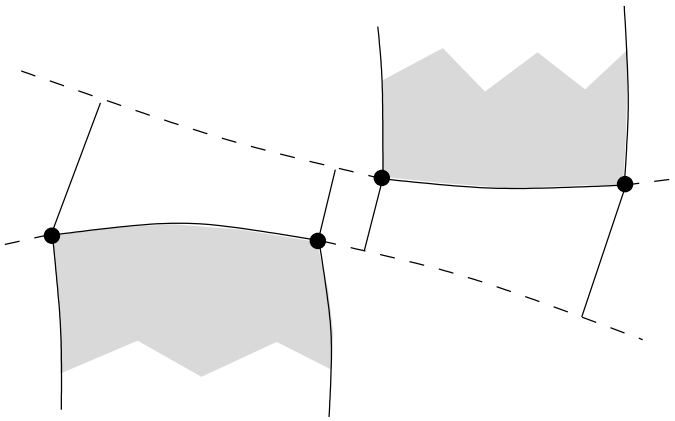

Figure 4.1: Every edge is part of a parabola. For two edges, if the projection of one edge on the parabola of the other edge is not part of the second edge, we exclude this pair of edges from our investigations.

Potential neighbours. It does not suffice to count the number of reference neighbourhoods missed by an algorithm; we also have to test for pairs of edges that an algorithm erroneously considers as neighboured. For a fair test, we do not examine all pairs of edges, but only those that are not obviously wrong. For a more precise explanation, recall that each parametric surface patch is bordered by its trimming curves. Each edge is a segment of such a parameterized curve. The segment is defined by a parameter interval of the parameter of the curve. If the projection of another edge onto this curve does not intersect with this interval, we exclude this pair of edges from our investigations (Figure 4.1).

Distance measures. We have examined two kinds of distance measures. The first one is represented by the distance measure that computes the mean value of the distances between ten pairs of mutually opposite points. These points are equidistant according to the parameterization of the edge. We will call this distance measure *uniform*.

The distance measures of the second kind are *weighted*. They result from the insight that an engineer sets the end points of an edge manually, whereas the interior of the edge is interpolated automatically. Therefore, the end points might be more precisely positioned than the interior of the edge. We measure the distance of two edges between three pairs of points. Like in Figure 4.2, the distances between the end points are included with the same

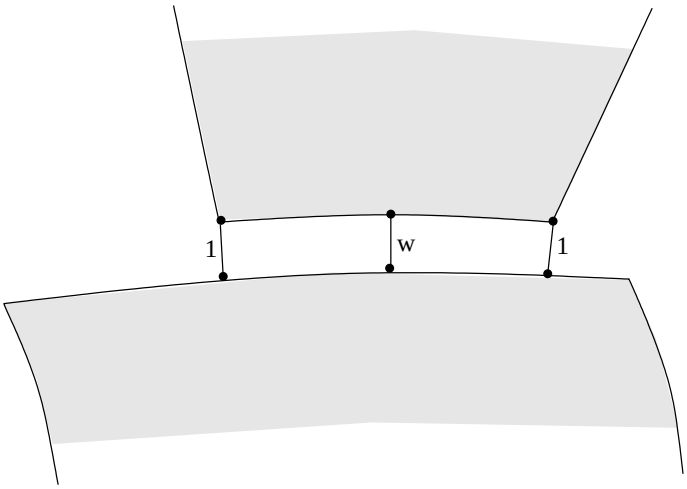

Figure 4.2: The weighted distance between two edges is the sum of three distances: the distances between the end points and the distance in the middle of a segment. The latter is multiplied with the weight w before the addition.

weight 1 for sake of symmetry. We refer to the individual distance measures by the weight of the third distance, which is the only difference between them. We will discuss at the end of this section, why we think that these distance measures are good representatives.

Examined minima. For each distance measure and workpiece we produced a diagram like Figure 4.3. The threshold value varies on the abscissa (scaled to fit in the diagram). The decreasing curve shows the number of reference neighbours that are missed by using the specified threshold value, whereas the increasing line documents the number of non-neighboured pairs of edges that are falsely found.

There are mainly two points of special interest in these diagrams. The first one is the total minimum number of faults – reference neighbours that are missed plus false neighbourhoods. The second interesting number is the minimum of wrong neighbours subject to the condition that all reference neighbourhoods are found. This minimum is more important than the first one, if one wants to find an overestimation of the neighbourhoods to fulfill the requirements for Theorem 9. Roughly speaking, an overestimation of a neighbourhood set is a neighbourhood set that contains at least the given neighbours. A more precise definition is given by:

Definition 5 (overestimation)
Let (N, D, O) and (N', D', O') be two feasible outputs. We say that (N, D, O) is an overestimation of (N', D', O'), if $N' \subset N$, and for all $(e_1, e_2) \in N'$ we have $D'(e_1, e_2) = D(e_1, e_2)$ and $O'(e_1, e_2) = O(e_1, e_2)$.

Let p be a point of an edge of a workpiece and (N', D', O') be an overestimation of (N, D, O). It is then easy to see that $\deg_{(N',D',O')}(p) \geq \deg_{(N,D,O)}(p)$.

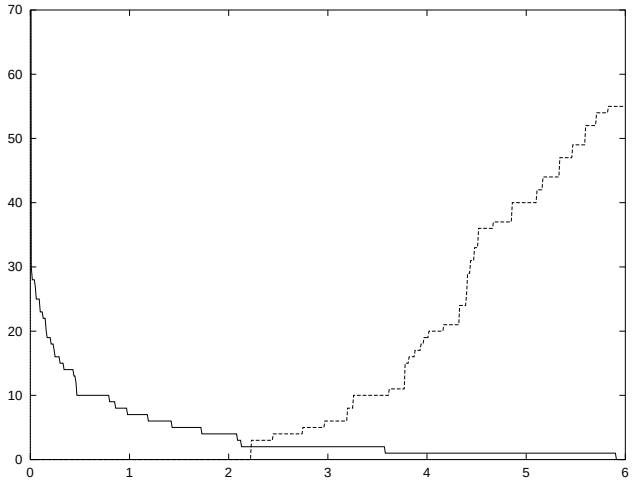

Figure 4.3: The number of reference neighbours that fail (decreasing line) and the number of pairs of edges that would be considered erroneously as neighboured (increasing line) for workpiece 8 and the uniform distance measure.

For each distance measure and each workpiece, we determined both minima. This is what we meant above by claiming to implicitly assume an algorithm that always finds the best threshold value for every workpiece.

Relative error. Our experiments have shown that estimating the relative error with respect to geometric attributes does not produce better results than using the absolute error. For a meaningful visualization of the results, the estimation

$$\frac{\text{distance}}{\text{scaling}} \leq \text{threshold}$$

has been replaced by

$$\text{distance} \leq \text{threshold} \cdot \text{scaling}.$$

In other words, we scale the threshold value instead of the distance function.

We have then determined all feasible threshold values for the (unscaled) uniform distance measure for every edge:

- The minimal feasible threshold value is the maximum of the distances to all reference neighbours.
- The maximal feasible threshold value is the minimum of the distances to all other edges.

A scaled threshold value for an edge between these bounds separates the correct neighbours of this edge from its false neighbours.

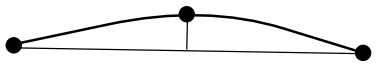

Figure 4.4: The *curvature* of an edge is measured by the distance of the supporting point in the middle of the edge to the line through the points at the ends of the edge.

We have examined the following four scalings:

1. the length of the edge,
2. the perimeter of the mesh element,
3. the area of the mesh element, and
4. the deviation of the edge from the straight line through the end point of the edge as shown in Figure 4.4. We call this distance the *curvature* of the edge.

For every scaling and workpiece, we have generated a diagram. In such a diagram, for every edge a vertical interval occurs according to the lower and upper bound of the threshold. The intervals are positioned on the abscissa according to the value of the scaling for the particular edge. (An example of such a diagram is shown in Figure 4.5.)

Our goal is to decide whether there is a reasonably simple interrelation between geometric characteristics and feasible threshold values. In our opinion, a suitable visualization produces the most convincing arguments about the complexity of a possible interrelation – a function that passes through all intervals. In particular, we determined the curve with minimal length in the set of continuous functions that pass between the permitted bounds.

This function with minimal length can be constructed automatically: the upper and lower bounds for each appearing value generate a polygon in the two dimensional plane. We seek for the shortest path from the leftmost segment to the rightmost one. According to [6] Section 8.2.1, a shortest path in a polygon is a subpath of the visibility graph of the vertices of the polygon. Up to a slightly different handling by replacing start and end points by segments, this solves our problem.

4.2 Computational Results

Absolute errors. We will first discuss the quality of the output for distance measures that represent the absolute error estimation. All of these distance measures failed for some workpieces. Furthermore, none of the distance measures turned out to be generally preferable over the others. The minima according to Section 4.1 are listed in Table 4.1 for individual distance measures and workpieces.

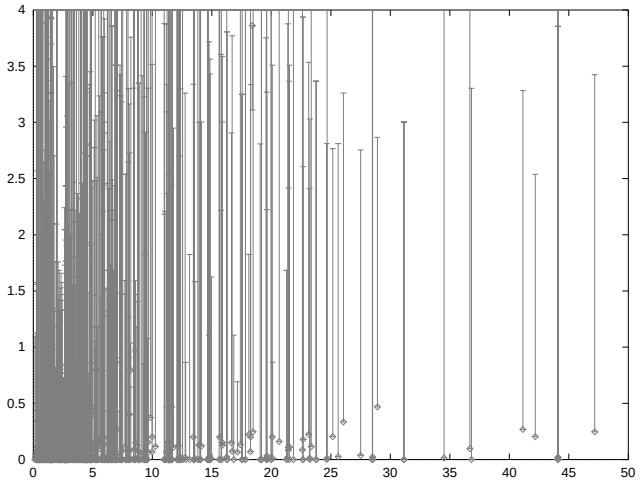

Figure 4.5: The possible threshold values for workpiece 12. Each interval shows the upper and the lower bound for one edge. Their position on the abscissa is the length of the edge.

Number of supporting points. We will first compare the uniform distance measure to a distance measure with only three measured distances, two at the end points and one in the middle of the edge. The results for the latter are listed in the column labeled "weighted 1.0". By comparing the two mentioned columns, the reader sees that the more fine-grained uniform distance measure does not lead to a better result. This realization justifies furthermore why we did not consider more than three supporting points for weighted distance measures.

Weighted distances. We have chosen four different values for the weight in the middle of the edge. The search for a favorable weight appears fruitless: for some workpieces lower threshold values are better, as for others higher ones are preferable. In most cases, however, the choice of the weight influences the quality of the output only marginally: the results are almost the same for all four weights.

Relative errors. We will now present the upper and lower bounds for the threshold value and discuss a possible interrelation with scaling values of the mesh elements. In the data that is available to us, we have even found some cases where the upper bound was smaller than the lower bound. To proceed with our investigations, we increased the upper bound to match the lower one in those cases in which no feasible threshold value exists at all.

Appendix A contains diagrams for all four scalings. They are typical in the sense that all of these functions "zigzag" irregularly, if there is no feasible unscaled threshold value for the workpiece. There are not even two functions that reveal any resemblance. These and all other diagrams suggest that probably there is not any usable interrelation between a suitable threshold value and any of the examined scalings.

No.	elements	neighbours		uniform		weighted		weighted		weighted		weighted	
		ref.	pot.			0.0		0.5		1.0		1.5	
1	34	55	1025	4	7	4	4	4	4	4	4	4	4
2	103	210	7860	1	3	0	0	0	0	0	0	0	0
3	295	676	35134	13	57	12	33	13	34	13	40	12	41
4	68	128	2304	0	0	0	0	0	0	0	0	0	0
5	251	573	23338	0	0	0	0	0	0	0	0	0	0
6	156	321	11525	26	216	18	72	18	72	20	72	22	72
7	342	823	48689	7	449	7	27	5	35	5	33	5	37
8	131	222	8669	2	55	0	0	0	0	1	5	2	11
9	530	1205	129596	0	0	0	0	0	0	0	0	0	0
10	608	1280	198055	0	0	0	0	0	0	0	0	0	0
11	24	40	567	0	0	0	0	0	0	0	0	0	0
12	179	409	20537	12	978	9	749	8	831	9	853	11	889
13	154	321	28860	3	3	3	3	2	2	2	2	2	3
14	237	446	29072	4	10	2	13	2	13	2	13	4	13
15	158	341	19085	0	0	3	3	0	0	0	0	0	0
16	156	261	17695	4	101	35	293	27	193	7	76	4	268

Table 4.1: Minimal number of wrong neighbourhoods for different distance measures. The weighted distance measure is determined according to Figure 4.2. For each distance measure, the first column lists the minimal number of false neighbourhoods – missing reference neighbours plus additional false neighbours. The second column contains the minimal additional false neighbours, provided that all reference neighbours are found.

Representative distance measures. For performing our tests we had to choose different distance measures. We think that our distance measures are good representatives and include the most natural ones. It is obvious that the distance measure should rely on a metric of the three dimensional space. The metric has to be invariant under translation and rotation of the workpiece. Since the algorithm should produce the same result for a scaled workpiece, the distance must be scalable, and thereby only those metrices that are derived from a norm are reasonable. Since all of those metrics are equivalent,[1] we selected the Euclidean distance as a representative. A natural extension of the Euclidean distance between two points to a distance measure of edges is the (discrete approximation of the) area between the edges.

In the computational study, we considered some related distance measures: other scalings than the length were treated, and different weights were used. Nevertheless, our results provide strong arguments that such changes do not improve the situation.

[1] i.e. for two norms $||\cdot||$ and $||\cdot||'$ there exist $c_1, c_2 \in \mathbb{R}_{>0}$ such that for all $x, y \in \mathbb{R}^3$: $||x-y|| \leq c_1 \cdot ||x-y||' \leq c_2 \cdot ||x-y||$

Of course, one might argue that a significantly more complicated and less natural distance measure could solve the problem. However, the conclusions from the empirical results turned out not to be too sensitive to variations of the distance measure. Switching to another distance measure can improve or impair the quality of the output for a single workpiece, but – probably – on average the result does not change dramatically.

Chapter 5

The Algorithm

5.1 Description

In the previous chapter we have seen that a pure estimation of the absolute or relative error will not solve the problem. This chapter presents an algorithm that incorporates some discrete techniques to overcome this problem. Furthermore, we seek for an overestimation of the neighbourhood set (Definition 5). In other words, we do not want to miss any real neighbourhood, whereas we admit wrong candidates. This will allow us – after proving Theorem 9 below – to easily verify the result for an instance.

The presented algorithm consists of several stages, which we consider one after the other in the individual subsections. The overall outline of the algorithm is as follows:

A. Find an overestimation of the set of real neighbourhoods.

B. Identify and remove as many neighbourhoods as possible which are obviously wrong.

More precisely, the algorithm proceeds in five stages. The first two stages determine an overestimation (N, D, O), the next three stages remove wrong neighbourhoods. Here are the individual stages in detail:

A. Stage 1: Determining an overestimating set $\hat{N}^{(1)}$ of neighboured pairs of *mesh elements*.

A. Stage 2: Determining an overestimating set $N^{(2)}$ of neighboured *edges*, their orientation $O^{(2)}$, and the domain $D^{(2)}$ where the edges in $N^{(2)}$ are neighboured.

B. Stage 3: Detecting and resolving conflicts between different candidate pairs of edges in $N^{(2)}$ which belong to the same pair of mesh elements.

B. Stage 4: Detecting and removing candidate pairs of segments which are highly implausible.

B. Stage 5: Reducing $N^{(4)}$ to a transitive relation $N^{(5)}$.

5.1.1 Stage 1: Neighboured Mesh Elements

In the first stage, we compute a set of pairs of mesh elements: the relation $N \subset E \times E$ of a neighbour set induces a relation $\hat{N} \subset M \times M$ of neighboured mesh elements. We first try to find $\hat{N}$. Since we want the algorithm to construct an overestimation of the neighbourhood set, we try to overestimate the actual pairs of neighboured mesh elements in this stage.

The general approach is as follows: we define a set $\ell_1, \ldots, \ell_\kappa$ of straight lines in $\mathbb{R}^3$ that pass through the origin $(0, 0, 0)$, a set $f_1, \ldots, f_\mu : \mathbb{R}^3 \to \mathbb{R}^3$ of congruential rotation functions, and a discrete threshold value $\theta \in \mathbb{N}$. For $i \in \{1, \ldots, \kappa\}$ and $j \in \{1, \ldots, \mu\}$, we compute the orthogonal projection of each mesh element onto each straight line $f_j(\ell_i)$. A pair of mesh elements is a candidate, i.e. added to $\hat{N}^{(1)}$, if for all $j \in \{1, \ldots, \mu\}$, the projections of these two mesh elements overlap on at least θ of the lines $f_j(\ell_1), \ldots, f_j(\ell_\kappa)$.

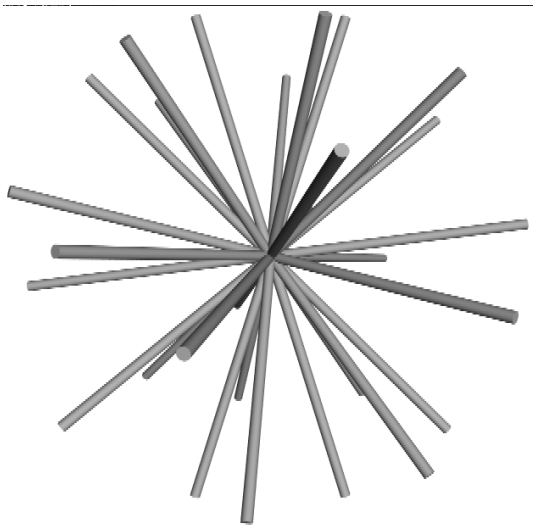

Figure 5.1: The 15 straight lines that we used in our computational experiments. All mesh elements of a workpiece were projected on five differently rotated copies of these lines.

For our computational experiments, we used $\kappa = 15$, $\mu = 5$, and $\theta = 10$. This choice is justified by experience: a significantly smaller or larger value of κ does not seem to allow an appropriate threshold value θ; the threshold value seems to be best chosen significantly larger than $\kappa/2$ and significantly less than κ; finally, a larger value of μ does not seem to make a difference and hence results in a waste of time.

We applied evolution strategies to choose $\ell_1, \ldots, \ell_{15}$ and $f_1, \ldots, f_5$. The straight lines $\ell_1, \ldots, \ell_{15}$ were determined using the following objective function: for each triple, ℓ_i, ℓ_j, and ℓ_k, we determine the angle α_{ijk} between ℓ_k and the plane spanned by ℓ_i and ℓ_j. Let β_{ij}

denote the fourth smallest value α_{ijk} over all $k = 1, \ldots, 15$, $k \neq i, j$. Then the value of the objective function is defined as $\min\{\beta_{ij} : i, j = 1, \ldots, 15, i \neq j\}$. This objective function was chosen in view of Theorem 11. More specifically, the evolution strategy yielded a set of straight lines such that the following fact, on which Theorem 11 in Sect. 5.2 relies, was achieved:

Fact 6
For every pair ℓ_{i_1}, ℓ_{i_2}, there are at most three straight lines ℓ_{i_3}, $i_3 \neq i_1, i_2$, such that the angle between ℓ_{i_3} and the plane spanned by ℓ_{i_1} and ℓ_{i_2} is 18 degrees or smaller.

To determine $f_1, \ldots, f_5$, the objective function of the evolution strategy was the minimum of all $\binom{15 \cdot 5}{2}$ angles between pairs of straight lines $f_j(\ell_i)$.

5.1.2 Stage 2: Neighboured Edges

First we compute a set of pairs of edges, which are intended to overestimate the real neighbourhoods of edges of mesh elements. Let $(m_1, m_2) \in \hat{N}^{(1)}$ be one of the pairs of mesh elements determined in the first stage, let $e_1^1, \ldots, e_k^1$ be the edges of m_1, and let $e_1^2, \ldots, e_l^2$ be the edges of m_2. The set $N^{(2)}$ of candidate pairs of edges consists of all (e_i^1, e_j^2) for all such pairs (m_1, m_2). For $(e_1, e_2) \in N^{(2)}$, we approximately compute the end points $D^{(2)}(e_1, e_2)$ of the domains where e_1 and e_2 are neighboured. This is simply done by projecting e_1 onto e_2 and vice versa. Furthermore, the orientation $O^{(2)}(e_1, e_2)$ is determined. Let p_i, q_i denote the points of e_i that are the images of the parameter intervals in $D^{(2)}(e_1, e_2)$. There are two possibilities to match these end points: p_1 with p_2 and q_1 with q_2, or p_1 with q_2 and q_1 with p_2. The orientation $O^{(2)}(e_1, e_2)$ is chosen to minimize the sum of the distance between the matching end points. Practical results show that this criterion is the right choice for all neighboured edges.

Obviously, if the first stage overestimates the set of pairs of neighboured mesh elements, then this stage overestimates the set of pairs of neighboured edges.

5.1.3 Stage 3: Conflicts between Neighbours I

Again let $(m_1, m_2) \in \hat{N}^{(1)}$ be a pair of mesh elements determined in the first stage. Consider the pairs (e_i^1, e_j^2) of segments in the output of the second stage such that one segment belongs to m_1 and the other one to m_2. Evidently, this set is much too large. In the third stage, we will reduce, for every such pair (m_1, m_2), this set of pairs of sides to an overestimation which comes much closer to the real neighbourhoods. If we knew that at most one pair of sides of m_1 and m_2 was really neighboured, then we could choose the pair which is geometrically "most plausible."

However, Figure 3.4 demonstrates that more than one pair of sides may be incident. Therefore, at this stage, we are much more conservative in removing pairs of sides from the initial list of candidates. The algorithm relies on the following assumption:

Assumption 7
No two segments of the same mesh element are neighboured.

The heuristic rationale behind Assumption 7 is as follows. From the intuitive description of the output in the Introduction, recall the heuristic one-to-one correspondence between the points on two neighboured edges. Suppose that two edges, e_1 and e_2, of the same mesh element m are neighboured, and let $p_1 \in e_1$ and $p_2 \in e_2$ be two points which correspond to each other according to $D^{(2)}$ and $O^{(2)}$ (cf. Definition 4). Then the geodetic line from p_1 to p_2 on m is the graph of a polynom with maximum width at p_1 and p_2. Hence, if p_1 and p_2 are to be regarded as one point, the width of this parabola is to be regarded as zero, which means that the mesh element must be regarded as being helplessly degenerate. In such a case, it is even preferable that the algorithm produces a wrong output, because then a well-prepared graphical visualization (cf. Theorem 9) gives the designer a chance to identify and correct this degenerate situation.

To explain the use of Assumption 7 in this stage, assume that (e_i^1, e_j^2) and (e_i^1, e_k^2) are two of the pairs of edges constructed in the previous stage, that e_j^2 and e_k^2 belong to the same mesh element m_2, but $e_j^2 \neq e_k^2$. Because of Assumption 7, this is an impossible situation, and since we assume that the candidate list overestimates the real neighbourhoods, we have to resolve this conflict by removing one of these conflicting pairs. More specifically, only the "most plausible" pair is included in $N^{(3)}$ from the list of candidates in $N^{(2)}$. The plausibility of a pair of edges is measured according to Definition 8 below. The practical results, which are presented in Section 5.4, show that this comparison of plausibilities does not seem to cause numerical problems. In particular, none of the reference neighbours is eliminated by this choice.

In the following, $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^3$. Moreover, for $p \in \mathbb{R}^3$ and for a finite, closed segment s of a parabola in $\mathbb{R}^3$, $d(p, s)$ denotes the minimum Euclidean distance from p to any point on s. Moreover, let $p(s)$ be the internal point of s "in the middle" (more precisely, in the middle of the parameter values of the end points with respect to the parameterization of the parabola). Roughly speaking, Definition 8 is intended to approximately relate the distances of end points of two segments, the distance between the "middle points" of the segments, and the lengths of the segments to each other.

Definition 8 (weighted distance and plausibility)
Let (e_1, e_2) be a pair of edges and (s_1, s_2) the segments that are induced by $D^{(2)}(e_1, e_2)$. Let denote h_i, t_i the end points of s_i so that h_1 corresponds to h_2 and t_1 corresponds to t_2 according to $O^{(2)}(e_1, e_2)$. Then, the weighted distance *$w(e_1, e_2)$ of (e_1, e_2) is defined as*

$$w(e_1, e_2) := \|h_1 - h_2\| + \|t_1 - t_2\| + \frac{1}{3}d(p(s_1), s_2) + \frac{1}{3}d(p(s_2), s_1)$$

where $p(s_i)$ is the point in the middle of the segment s_i according to its parameterization. Furthermore, the plausibility is defined as

$$\frac{w(e_1, e_2)}{\|h_1 - t_1\| + \|h_s - s_2\|}$$

Empirically, this definition of plausibility has turned out to be more appropriate than simpler definitions. This might be due to the following insights. First of all, Figure 1.2 demonstrates that it does not suffice to merely take the distances of the end points of two segments: two segments in Figure 1.2 which form a ring would be regarded as a highly plausible pair, which is absurd. To overcome this problem, we take the distance of two segments in the "middle" into account. For reasons of symmetry, we consider both $d(p(s_1), s_2)$ and $d(p(s_2), s_1)$. However, it is our general experience that the end points of two neighboured segments are typically much closer to each other than the points somewhere in between. Figure 1.3 shows a drastical example of this fact. Therefore, the contribution of $d(p(s_1), s_2)$ and $d(p(s_2), s_1)$ to the plausibility of the pair (s_1, s_2) should be relatively small. It seems that 1/3 is a good compromise in practice.

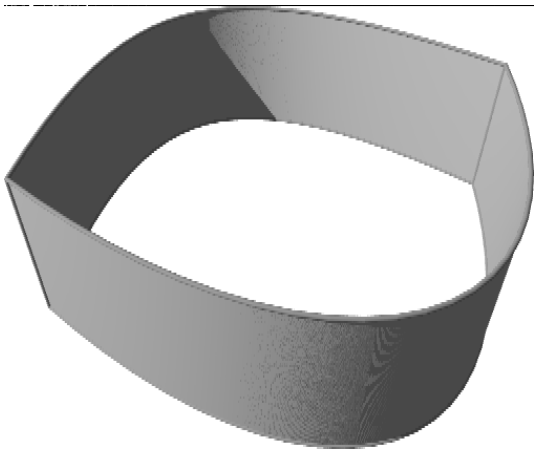

Figure 5.2: Two mesh elements of workpiece 15 (see also Figure 1.3). Their upper edges are not neighboured, although the respective end points coincide.

The output of this stage is $(N^{(3)}, D^{(3)}, O^{(3)})$ where $D^{(3)}$ and $O^{(3)}$ are the restrictions of $D^{(2)}$ and $O^{(2)}$ to $N^{(3)}$. This stage is a crucial prerequisite for the procedure in Section 5.1.5. See the additional remarks at the end of that section.

5.1.4 Stage 4: Conservative Error Corrections

The list of pairs of segments constructed in the previous stage is further reduced by estimating the relative error. This stage is the only "traditional" part of the algorithm, where no discrete information is used. However, the individual tests are listed for sake of completeness.

All checkers are parameterized, and we chose all parameters such that the respective check is *very* conservative. In fact, we have determined all of these parameters such that all correct incident pairs of segments in all real-world instances available to us pass each

of these checks, and none of these pairs comes close to the limits. A candidate pair of segments is dropped if at least one of these tests fails. Table 5.1 shows the results of these checkers.

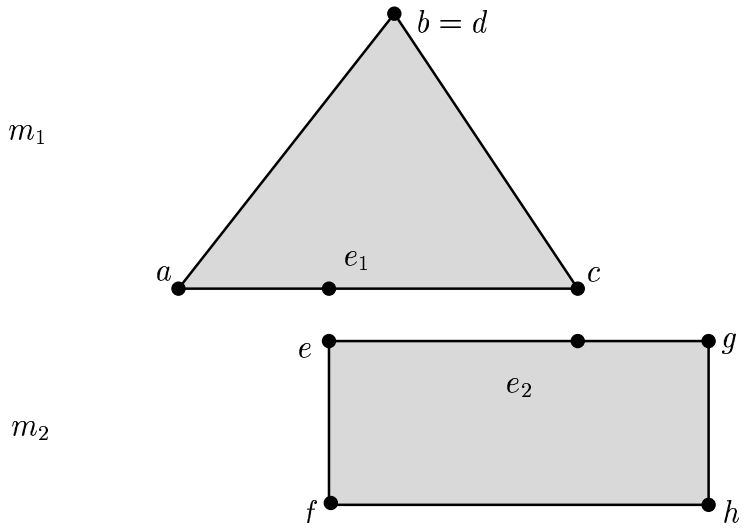

Figure 5.3: The situation in the second conservative checker.

We also experimented with other checkers. Those that did not improve the result were excluded from further investigations. The following checkers "survived" these experiments:

1. The first checker eliminates (at least) the pairs of edges in $N^{(3)}$ that are not potential neighbours in the sense of Section 4.1. Imagine that each of e_1 and e_2 with $(e_1, e_2) \in N^{(3)}$ is continued to the infinite quadratic curve as depicted in Figure 4.1. If the projection of e_1 onto the continuation of e_2 covers less than two percent of e_2, or vice versa, then the pair (e_1, e_2) is dropped.

2. Let $a, \ldots, h$ denote the corners of the sides of m_1 and m_2 next to the sides of e_1 and e_2 as shown in Figure 5.3 (two points, b and d or f and h, are identical, if the corresponding mesh element is a triangle). The weighted distance (cf. Definition 8) must not exceed $0.28 \cdot \max\{\|a - b\| + \|c - d\|, \|e - f\| + \|g - h\|\}$.

3. Let s_1 and s_2 denote the segments on e_1 and e_2 that are induced by $D^{(3)}$. The cosine of the angle between the two straight lines through the end points of s_1 and through the end points of s_2 must be at least 0.875 (i.e. the angle is less than 29 degrees).

4. The cosine of the angle between the planes spanned by s_1 and s_2 must be at least 0.99 (i.e. 8.1 degrees; this test is not applied, if one of s_1 or s_2 is straight or close to a straight line, in which case the plane is not unique).

5. Let $p_1, q_1 \in e_1$ and $p_2, q_1 \in e_2$ the end points such that p_1 corresponds to p_2 and q_1 corresponds to q_2. Then

$$\max\{\|t_1 - t_2\|, \|h_1 - h_2\|\} \leq \min\{\|t_1 - t_2\|, \|h_1 - h_2\|\} + \frac{1}{5}(\|t_1 - h_1\| + \|t_2 - h_2\|)$$

must be fulfilled.

6. First we compute approximate normal vectors, v_1 and v_2, for m_1 and m_2, namely normal vectors of planes which approximate m_1 and m_2 and contain the end points of the edges e_1 and e_2 (see Figure 5.4). Let p_1, p_2, p_3, p_4 (resp. q_1, q_2, q_3, q_4) denote the orthogonal projections of h_1, h_2, t_1, and t_2 onto the straight line $\mathbb{R} \cdot v_1$ ($\mathbb{R} \cdot v_2$). This test succeeds if and only if none of $\|p_1 - p_2\|$, $\|p_3 - p_4\|$, $\|q_1 - q_2\|$, and $\|q_3 - q_4\|$ exceeds $\frac{1}{5}(\|h_1 - t_1\| + \|h_2 - t_2\|)$. This test is not applied, if the cosine of the angle between v_1 and v_2 or the cosine of the angle between $h_1 - t_1$ and $h_2 - t_2$ is less than 0.9 (i.e. 25.8 degrees).

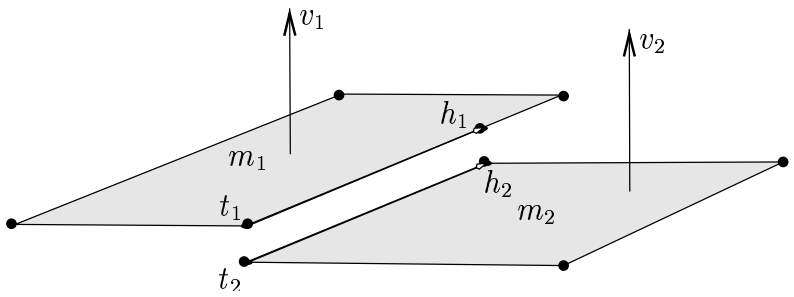

Figure 5.4: The situation in the sixth conservative checker. The behind rationale this test is to eliminate false neighbourhoods between parallel mesh elements.

The checkers are used to produce a new intermediate result $(N^{(4)}, D^{(4)}, O^{(4)})$ from $N^{(3)}$: a pair of edges $(e_1, e_2) \in N^{(3)}$ is included in $N^{(4)}$ if and only if it passes all checkers. $D^{(4)}$ and $O^{(4)}$ are the restrictions of $D^{(3)}$ and $O^{(3)}$ to $N^{(3)}$

5.1.5 Stage 5: Conflicts between Neighbours II

In this stage, the set $N^{(4)}$ is reduced to a new intermediate result $N^{(5)}$. More precisely, the set $N^{(5)}$ is an inclusion-maximal transitive subset of $N^{(4)}$, that is, for (e_1, e_2), $(e_1, e_3) \in N^{(5)}$, we also have $(e_2, e_3) \in N^{(5)}$. So consider (e_1, e_2), $(e_1, e_3) \in N^{(4)}$ such that $(e_2, e_3) \notin N^{(4)}$, To resolve this conflict, either (e_1, e_2) or (e_1, e_3) must be dropped. Here is our rule for

deciding which one to drop:

1. If there is $(e_3, e_4) \in N^{(4)}$ such that e_4 belongs to the same mesh element as e_2, and $e_4 \neq e_2$, then we drop (e_1, e_3);

2. Otherwise, we drop the less plausible pair (cf. Definition 8).

Figure 5.5 demonstrates the rationale behind the first rule: let us assume $(e_1, e_3) \in N^{(4)}$. If $N^{(4)}$ actually overestimates the real neighbourhoods as intended, we have (e_1, e_2), (e_1, e_3), (e_3, e_4) in $N^{(4)}$. Due to Stage 5.1.2, we have $(s_1, s_4) \notin E_1$, because (e_1, e_2) and (e_3, e_4) are certainly more plausible. Hence, the first rule applies and removes the wrong neighbourhood (e_1, e_3), even if, for whatever geometric reasons, (e_1, e_3) is more plausible than (e_1, e_2) according to Definition 8.

By application of this rule the problem depicted in Figure 3.4 can be solved: the first stages can safely include the neighbourhoods around the gap on the left and on the right side, because the one on the right side will certainly be removed in this stage. This results in correct neighbourhoods on both sides.

Unfortunately, the first rule cannot be extended to more than one mesh element. Figure 5.6 shows that two edges can be neighboured although there are two mesh elements between them.

Remark. The first rule applies quite often. There are configurations in real-world instances where the first rule applies and drops the more plausible pair (unfortunately, the crucial properties of these configurations cannot be shown in a two-dimensional figure). However, in all of these cases, the first rule was right. This is not surprising, because the first rule makes use of discrete, structural information, whereas the second rule relies on mere numerical computations.

At first glance, having two rules to remove a neighbourhood in a conflict might seem self-contradictory: we rely on the plausibility check in the third stage, but do not really trust the plausibility check in the fifth stage. However, the situation is different: in the third stage, the plausibility check is applied to compare two candidates which belong to the same pair of mesh elements. It seems that the comparison of plausibilities is reliable in this case.

The output $(N^{(5)}, D^{(5)}, O^{(5)})$ of the algorithm is now *transitive*. In Section 2.2 we mentioned that we implicitly assume it to be *symmetric*. Up to the diagonal in $E \times E$ the relation "correspondence according to $(N^{(5)}, D^{(5)}, O^{(5)})$" therefore induces an equivalence relation of the points. That means that the output is a feasible neighbourhood.

5.2 Accuracy

As mentioned in Section 3.3, the result of an instance is inherently subjective to the designer of the workpiece. That's why the correctness of the result cannot be proven mathematically.

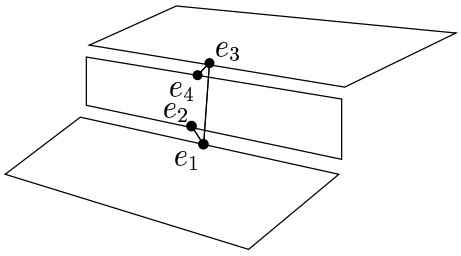

Figure 5.5: Edge e_1 is neighboured with edge e_2, but not with e_3 because of the quadrilateral in between.

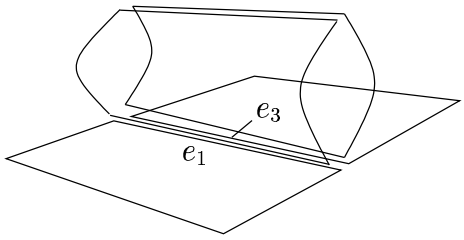

Figure 5.6: Edge e_1 is neighboured with edge e_3 although there are two mesh elements "in between".

However, we aim at an output (N', D', O') in which the errors can be easily checked by eye. The correctness check is simple: we use different colours, $c_1, c_2, c_3, \ldots$, to visualize the number of neighbourhoods. For every edge e each point $p \in e$ is assigned the colour $c_d(p)$ where $d(p)$ denotes the degree of the point p as defined in Definition 4. The check by eye succeeds if and only if each point on the border is assigned the colour that is expected by a human observer who has the result (N, D, O) in mind.[1] We will show that any *overestimation* (N', D', O') of (N, D, O) fulfills this condition:

Theorem 9
Let (N, D, O) and (N', D', O') be feasible outputs and let (N', D', O') be an overestimation of (N, D, O). We have $(N', D', O') = (N, D, O)$ if and only if

$$\deg_{(N',D',O')}(p) = \deg_{(N,D,O)}(p)$$

for all $p \in e$ and every edge $e \in E$ of the workpiece.

Proof: The *only-if* part is trivial, because $\deg_{(N',D',O')} = \deg_{(N,D,O)}$ if the neighbourhood sets (N', D', O') and (N, D, O) are equal. So consider the *if* part. For sake of a contradiction we assume that $(N', D', O') \neq (N, D, O)$. It follows from the definition of "overestimation" that for every p_2' to which p_1 corresponds according to (N, D, O), p_1 also corresponds to p_2' according to (N', D', O') (cf. Definition 4). As a consequence we have

$$\deg_{(N',D',O')}(p_1) \geq \deg_{(N,D,O)}(p_1)$$

Since $(N', D', O') \neq (N, D, O)$ and (N', D', O') is an overestimation of (N, D, O) there exists a pair $(e_1, e_2) \in N'$ with $(e_1, e_2) \notin N$. Choose $p_1 \in e_1$ and $p_2 \in e_2$ such that p_1 corresponds to p_2 according to (N', D', O'). Obviously p_1 does not correspond to p_2

[1] In practice, it is not possible to colour an insulated point on an edge. Nevertheless, the user of the algorithm can visually check the result, because their occurrence does not matter for practical results.

according to (N, D, O), because (e_1, e_2) is not in N. This leads to

$$\deg_{(N',D',O')}(p_1) > \deg_{(N,D,O)}(p_1),$$

which is a contradiction. □

Corollary 10
Since the algorithm produced an overestimation of $(\mathcal{N}, \mathcal{D}, \mathcal{O})$ for every real-world instance in Table 5.2, the errors in the solution to any of them can be found by eye.

From a practical point of view, Corollary 10 seems to be the best one can hope for. Obviously, there is no algorithm for which the correctness of the result can be proven, because it only exists in the mind of the engineer. It is therefore necessary to have a simple procedure to check the output by eye. Figure 5.7 shows an example of such a visualization, which makes it easy to verify the result.

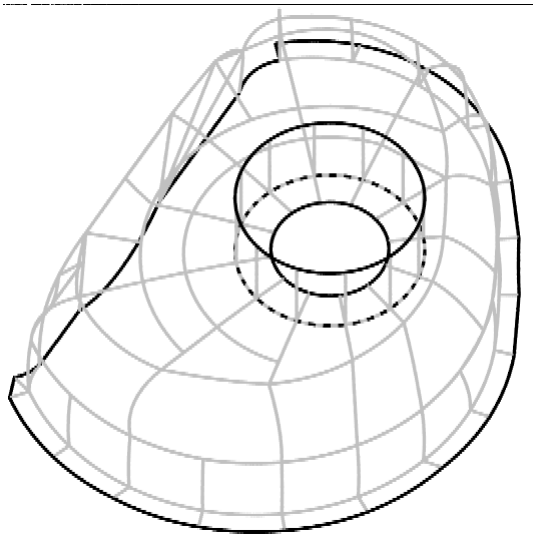

Figure 5.7: This is a part of workpiece 10. (The complete workpiece is shown in Figure B.10.) It has a colouring according to Corollary 10: the borders – the edges with degree 1 – are black. The edges with degree 2 are grey. The dashed line shows a branching with edges of degree 3.

Next we formulate sufficient conditions for the algorithm to deliver an overestimation of $(\mathcal{N}, \mathcal{D}, \mathcal{O})$. In Theorem 11 below, we will give a sufficient condition for the first stage to deliver an overestimation of the set of neighboured mesh elements. If the first stage produces an overestimation, then trivially the second stage provides an overestimation of $(\mathcal{N}, \mathcal{D}, \mathcal{O})$. The third stage is correct if no two segments of the same mesh element are neighboured (cf. Assumption 7). The conditions for the fourth stage are also obvious. The

fifth stage is correct if, in case of a conflict as described in Section 5.1.5, the plausibility distinguishes the correct neighbourhood from the wrong one.

In the following, we distinguish between points and directions in $\mathbb{R}^3$, and directions are indicated by arrows. Moreover, $\sphericalangle(\vec{u}_1, \vec{u}_2)$ denotes the angle between directions $\vec{u}_1$ and $\vec{u}_2$.

Theorem 11
For $(e_1, e_2) \in \mathcal{N}$, let $t_1 \in e_1$ and $t_2 \in e_2$ (resp. $h_1 \in e_1$ and $h_2 \in e_2$) be the mutually corresponding end points according to $(\mathcal{N}, \mathcal{D}, \mathcal{O})$. Furthermore, define $\vec{v}_1 := h_2 - t_1$ and $\vec{v}_2 := t_2 - h_1$. Finally, let α_0 denote the maximal angle so that for every pair ℓ_{i_1}, ℓ_{i_2}, there are at most three straight lines ℓ_{i_3}, $i_3 \neq i_1, i_2$, such that the angle between ℓ_{i_3} and the plane spanned by ℓ_{i_1} and ℓ_{i_2} is smaller or equal to α_0. The first stage delivers this pair of neighboured mesh elements, if for $(e_1, e_2) \in \mathcal{N}$, the angle $\sphericalangle(\vec{v}_1, \vec{v}_2)$ is at least $180° - \alpha_0$.

Proof: Assume for a contradiction that for $(e_1, e_2) \in \mathcal{N}$, the pair of mesh elements to which e_1 and e_2 belong is not delivered by the first stage, although $\sphericalangle(\vec{v}_1, \vec{v}_2)$ is greater than $180° - \alpha_0$. Then there is some $j \in \{1, \ldots, 5\}$ and there are pairwise different numbers $i_1, \ldots, i_6 \in \{1, \ldots, 15\}$ such that the projections of e_1 and e_2 do not overlap on any of the straight lines $f_j(\ell_{i_\mu})$.

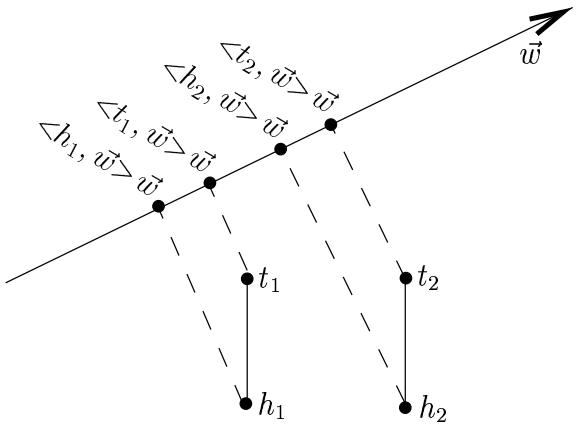

Figure 5.8: Proof of Theorem 11. The figure shows the first alternative in the formalization of the condition that the projections of e_1 and e_2 do not overlap.

Clearly, translating straight lines $f_j(\ell_i)$ in $\mathbb{R}^3$ does not change the result of the first stage. Hence, we may identify each straight line $f_j(\ell_i)$ with one of its two possible orientations. It is well known that $<p, \vec{w}>\vec{w}$ is the end point of the perpendicular of point $p \in \mathbb{R}^3$ dropped onto the straight line $\mathbb{R} \cdot \vec{w}$, $\vec{w} \in \mathbb{R}^3 \setminus \{0\}$. Hence, if the projections of s_1 and s_2 on $\vec{w}$ do

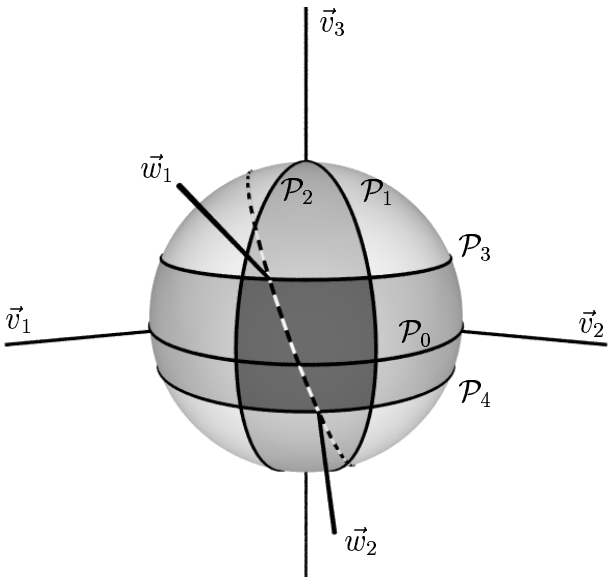

Figure 5.9: Proof of Theorem 11. The figure shows the unit sphere and its intersections with the planes $\mathcal{P}_0 - \mathcal{P}_4$. All points $\vec{w}_1, \ldots, \vec{w}_6$ are in the dark grey area. The dashed line is the intersection of the unit sphere with the plane spanned by $\vec{w}_1$ and $\vec{w}_2$. The angle between this plane and any point in the dark grey area is at most α_0.

not overlap, one of the following two alternatives is true (see Figure 5.8):

$$<h_2 - h_1, \vec{w}>, <t_2 - h_1, \vec{w}>, <h_2 - t_1, \vec{w}>, <t_2 - t_1, \vec{w}> \; > \; 0$$

or

$$<h_2 - h_1, \vec{w}>, <t_2 - h_1, \vec{w}>, <h_2 - t_1, \vec{w}>, <t_2 - t_1, \vec{w}> \; < \; 0\,.$$

In particular, the following fact holds in both cases:

> The projections of e_1 and e_2 onto direction $\vec{w} \in \mathbb{R}^3$ overlap, unless $\sphericalangle(\vec{v}_1, \vec{w})$ and $\sphericalangle(\vec{v}_2, \vec{w})$ are both greater than 90 degrees or both less than 90 degrees.

Without loss of generality, we assume in the following that $\sphericalangle(\vec{v}_1, \vec{w})$ and $\sphericalangle(\vec{v}_2, \vec{w})$ are both less than 90 degrees, if s_1 and s_2 do not overlap on $\vec{w}$. In particular, for $\mu \in \{1, \ldots, 6\}$, let $\vec{w}_\mu$ be the orientation of $f_j(\ell_{i_\mu})$ according to this rule. In the rest of this proof, we will assume that all directions are normed, that is $\|\vec{v}_1\| = \|\vec{v}_2\| = \|\vec{w}_1\| = \cdots = \|\vec{w}_6\| = 1$. Therefore, we can identify each direction with the corresponding point on the unit sphere (see Figure 5.9).

Let $\vec{v}_3$ be a direction orthogonal to $\vec{v}_1$ and $\vec{v}_2$. Without loss of generality, $\sphericalangle(\vec{w}_1, \vec{v}_3) = \min\{\sphericalangle(\vec{w}_i, \vec{v}_3) : i = 1, \ldots, 6\}$ and $\sphericalangle(\vec{w}_2, vecv_3) = \min\{\sphericalangle(\vec{w}_i, -\vec{v}_3) : i = 2, \ldots, 6\}$. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ denote the planes that contain the origin and are orthogonal to $\vec{v}_1$ and $\vec{v}_2$. Moreover,

$\mathcal{P}_3$ and $\mathcal{P}_4$ denote the planes that contain $\vec{w}_1$ and $\vec{w}_2$ and are parallel to the plane $\mathcal{P}_0$ spanned by $\vec{v}_1$ and $\vec{v}_2$.

Note that $\mathcal{P}_1$ and $\mathcal{P}_2$ decompose the unit sphere into four components: two components of width α_0, and two components of width $180° - \alpha_0$. In particular, the area of the unit sphere between $\mathcal{P}_3$ and $\mathcal{P}_4$ is decomposed in this way. Then all $\vec{w}_1, \ldots, \vec{w}_6$ belong to one of the components having width α_0. Now standard arguments from spherical trigonometry yield that the angle between any of $\vec{w}_3, \ldots, \vec{w}_6$ and the plane spanned by $\vec{w}_1$ and $\vec{w}_2$ is at most α_0. However, this contradicts the maximality of α_0. □

Remark. The real-world instances listed in Table 5.2 contain 33 correct neighbourhoods in total for which $\sphericalangle(\vec{v}_1, \vec{v}_2)$ is less than 162 degrees. So – theoretically – they could be missed by the algorithm, because for the used set of lines α_0 is 18° as stated in Fact 6.

5.3 Complexity

Let $m := |M|$ denote the number of mesh elements in the input. In the first stage, all mesh elements are projected onto $15 \cdot 5$ straight lines. For $i \in \{1, \ldots, 15\}$, $j \in \{1, \ldots, 5\}$, let $n_{i,j}$ denote the number of pairs of mesh elements whose projections overlap on $f_j(\ell_i)$. For $j \in \{1, \ldots, 5\}$, N_j denotes the tenth smallest value out of $n_{1,j}, \ldots, n_{15,j}$, and $n = \max\{|E|, \max\{N_j : j = 1, \ldots, 5\}\}$. For $e \in E$, d_e denotes the *degree* of e in $N^{(4)}$, which is defined as $\left|\left\{e' \in E \;\middle|\; (e, e') \in N^{(4)}\right\}\right|$. (For a point $p \in e$ it is possible that $\deg_{(N^{(4)}, D^{(4)}, O^{(4)})}(p) < d_e$.)

Theorem 12
The asymptotic complexity of the algorithm is $\mathcal{O}\left(m \log m + n \log n + \sum_{e \in E} d_e\right)$.

Proof: Consider the first stage. Projecting all mesh elements onto all straight lines $f_j(\ell_i)$ takes $\mathcal{O}(m)$ time. Next we sort the end points of the projections of all mesh elements on each $f_j(\ell_i)$ in the order of occurrence on $f_j(\ell)$, which takes $\mathcal{O}(m \log m)$ time. For each $j \in \{1, \ldots, 5\}$, we determine the values $n_{1,j}, \ldots, n_{15,j}$. It is easy to see that $n_{i,j}$ can be computed for every $f_j(\ell_i)$ in linear time by a single pass over all end points of projections in sorting order. Let $j \in \{1, \ldots, 5\}$, and let $f_j(\ell_{i_1}), \ldots, f_j(\ell_{i_{10}})$ be the ten straight lines out of $f_j(\ell_1), \ldots, f_j(\ell_{15})$ with the smallest values of $n_{i,j}$. For each pair of overlapping mesh elements on $f_j(\ell_{i_\mu})$, $\mu = 1, \ldots, 10$, we determine the number of straight lines $f_j(\ell_1), \ldots, f_j(\ell_{15})$ on which this pair overlaps, and we accept this pair, if the number of straight lines is at least $\Theta = 10$ for each j. Due to the specific definition of n, this requires $\mathcal{O}(n)$ time.

The pairs of edges to be considered in the second, third, and fourth stage is $\mathcal{O}(n)$, because there can be at most $4 \cdot 4$ pairs of edges for one pair of mesh elements. Clearly, for each pair of edges (e_1, e_2) the values $D^{(2)}(e_1, e_2)$ and $O^{(2)}(e_1, e_2)$ can be constructed and checked in constant time, and the claimed complexity follows for each of the three stages.

In the fifth stage, we sort all pairs $(e_1, e_2) \in N^{(4)}$ according to their plausibilities, which takes $\mathcal{O}(|n| \log |n|)$ time. Using this sorting, we can construct and sort all lists $A(e) = \left\{ (e_1, e_2) \in N^{(4)} \;\middle|\; e = e_1 \wedge e = e_2 \right\}$ for all $e \in E$ in $\mathcal{O}(n)$ total time. For a pair $(e_1, e_2) \in N^{(4)}$, let $\Delta(e_1, e_2)$ denote

$$\Delta(e_1, e_2) := \big\{ (e_1, e') \in A(e_1) \;\big|\; D_1(e_1, e_2) \cap D_1(e_1, e') \neq \emptyset \;\; \wedge \\ \forall (e_2, e') \in A(e_2) D_2(e_1, e_2) \cap D_1(e_2, e') = \emptyset \big\}$$

Since all sets $A(\cdot)$ are ordered according to the same key (namely the plausibility), all sets $\Delta(\cdot, \cdot)$ can be computed in $\mathcal{O}\left(\sum_{e \in E} d_e\right)$ total time.

The rest of the fifth stage consists of two phases: first we drop all pairs $(e_1, e_2) \in N^{(4)}$ to which the first rule in Section 5.1.5 applies; afterwards, all pairs to which the second rule applies. Whenever we drop a pair $(e_1, e_2) \in N^{(4)}$, we update all values $\Delta(e_1, \cdot)$ and $\Delta(e_2, \cdot)$ accordingly. Therefore, a pair (e_1, e_2) is a possible candidate for dropping if and only if $\Delta(e_1, e_2) \neq \emptyset$. Moreover, these updates ensure that the resulting equivalence relation $N^{(4)}$ is inclusion-maximal. Every item (e_1, e_2) in a list $\Delta(e_1, e_3)$ has a natural "mate" (e_1, e_3) in the list $\Delta(e_1, e_2)$. If all of these pairs of mates refer to each other, then updating all $\Delta(e_1, \cdot)$ and $\Delta(e_2, \cdot)$ in case (e_1, e_2) is dropped takes $\mathcal{O}(d_{e_1} + d_{e_2})$ time. In total, this adds up to $\mathcal{O}\left(\sum_{e \in E} d_e^2\right)$. In the first phase, we may scan all pairs (e_1, e_2) in arbitrary order. However, in the second phase, we scan them in increasing order of plausibility, which ensures that always the least plausible pair is dropped. This concludes the analysis of the fifth stage. □

As mentioned in the Introduction, m and n are strongly linearly correlated in the real-world instances available to us. In fact, although a particular value $n_{i,j}$ may be $\Theta(n^2)$ (for example, when all mesh elements are contained in the plane orthogonal to $f_j(\ell_i)$), $n_{i,j} \notin \mathcal{O}(n)$ seems next to impossible for the tenth smallest value out of $n_{1,j}, \ldots, n_{15,j}$. Therefore, it is reasonable to assume $N \in \mathcal{O}(n)$. Moreover, it might be evident that $M_1, \ldots, M_K \in \mathcal{O}(1)$ is a reasonable practical assumption.

Corollary 13
If $N \in \mathcal{O}(n)$ and $M_1, \ldots, M_K \in \mathcal{O}(1)$, the asymptotic complexity is $\mathcal{O}(n \log n)$.

5.4 Computational Result

We have implemented a variant of the algorithm in C++ on Sun workstations using the GNU compiler of the Free Software Foundation, Inc. For technical reasons, the implementation differs from the algorithm described in Section 5.1. For example, the intermediate results are not always constructed as data structures, but evaluated in a pipelining process to reduce the space requirements.

To give an impression of the impact on each stage, we have summarized the number of neighbours in Table 5.1. In the third stage a discrete inference rule is applied to remove pairs of edges. This rule turns out to be quite effective. In case of two triangles there are 9

#neighboured mesh elements after stage 1		80106	96879
#neighboured sides after stage 3		83460	96665
stage 4	#neighbourhoods passing test 1	13344	12294
	#neighbourhoods passing test 2	76855	92018
	#neighbourhoods passing test 3	44229	35812
	#neighbourhoods passing test 4	37190	50609
	#neighbourhoods passing test 5	83077	96665
	#neighbourhoods passing test 6	80757	8328
#neighboured sides after stage 4		8633	8328
#neighboured sides after stage 5		7180	7629
correct #neighboured sides		7050	7546

Table 5.1: The effects of the individual stages onto the result. The second column gives the total number for the first 15 instances in table 5.2, and the third column, the total number for the remaining instances. To compile these results, each checker from Section 5.1.3 was applied to all pairs of neighboured sides constructed in the third stage (in contrast, in the algorithm, these checkers form a pipeline).

possible pairs of edges, in case of a triangle and a quadrilateral there are 12 possible pairs of edges, and in case of two quadrilaterals there are 16 possible pairs of edges. However, after the third stage there rests on average about one pair of edges per pair of mesh elements.

Although the checkers in stage 4 are tuned to produce a good approximation, the fifth stage still removes more than 1000 pairs of edges (in summary).

Table 5.2 shows the results of a computational study. The study is based on a reference set of neighbourhoods as described in Section 4.1, which we regard as correct. The first 15 instances stem from our concrete application and consist mainly of quadrilaterals. The other instances have been collected from various public sources. Instances nos. 16-20 consist solely of plain mesh elements, which are bounded by straight sides. In these instances two neighboured edges are neighboured along the whole parameter interval. Moreover, instances nos. 17-20 are entirely composed of triangles. For each instance, this table gives the number of mesh elements in the input (mesh elements), the number of correct neighboured pairs of segments in the reference data (neighbours), and the number of pairs falsely delivered by our algorithm (wrong pairs); no correct pair was missing.

The average run time of the implementation of the algorithm taken over all of these instances is less than one minute on a Sparc Ultra 4000/5000 (incl. input/output). As mentioned in the Introduction, these run times are negligible, because the finite-element method takes orders of magnitude more time.

no.	mesh elements	neighbours	wrong pairs
1.	34	55	0
2.	103	210	3
3.	295	676	44
4.	68	128	2
5.	251	573	1
6.	156	321	0
7.	342	823	15
8.	131	222	0
9.	530	1205	2
10.	608	1280	4
11.	24	40	0
12.	179	409	2
13.	154	321	43
14.	237	446	0
15.	158	341	14
16.	156	261	73
17.	1242	1844	0
18.	1243	1786	0
19.	1025	1529	10
20.	1417	2126	0
Total:	8353	14596	213

Table 5.2: Computational results for real-world instances. The first 15 instances stem from the concrete application (mostly parts of traffic vehicles). The other instances have been collected from various public sources. Instances no. 16-20 consist solely of plain mesh elements, which are bounded by straight sides. In these instances two neighboured edges are neighboured along the whole parameter interval. Moreover, instances no. 17-20 are entirely composed of triangles. Images of the workpieces are shown in Appendix B. For each instance, this table gives the number of mesh elements in the input, the number of neighboured pairs of segments in the reference data, and the number of pairs of segments that were falsely identified by our algorithm as neighboured (the converse error, that is missing correct neighbourhoods, did not occur).

5.5 Parameter Intervals Revisited

The algorithm that has been presented and analysed in the previous section does a good job in determining the neighboured edges. In the second stage of the algorithm, we determined the domain $D^{(2)}(e_1, e_2)$ of a neighboured pair of edges (e_1, e_2) only approximately by dropping the perpendicular of the end points of e_1 on e_2, and vice versa. This is acceptable, because our experience suggests that the following stages are robust against small changes of $D(e_1, e_2)$. However, it does not reveal the full truth. It is sometimes preferable to alter the end points of the neighbourhoods slightly.

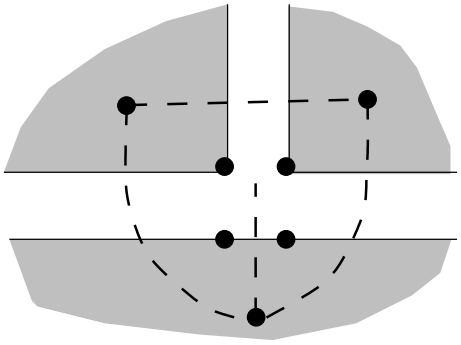

Figure 5.10: Dropping the perpendiculars of two equivalent points onto the same edge induces an unintended segment, which is not neighboured to any other edge.

The gaps between neighboured edges have a strong impact on $D(e_1, e_2)$: dropping perpendiculars from two corresponding points onto the same side induces a small segment between the end points of the two perpendiculars (see Figure 5.10). For the subsequent tasks in the CAD process it is usually better to work with modified domains $D(e_1, e_2)$, than being faced with these small segments. The proposed clean-up in this section essentially amounts to finding and eliminating them.

More precisely, we consider the end points of neighboured edges, but only those that are actually vertices of the respective edges. Then, the mean value of the positions of neighboured vertices – their center of gravity – is calculated. If an edge is neighboured to more than one other edge, all corresponding vertices of the other edges are included. Now, instead of using the real vertices, we drop the perpendicular from this point to every edge whose open interior contains an element of this equivalence class. This operation defines the final end points of the parameter interval and hence the final solution. Once again, we successfully applied a discrete technique in using the neighbours N for the determination of their domains D.

Chapter 6

Conclusion

Error estimation. The common approach of using a distance measure and a global threshold value seems to fail for a variety of unscaled distance measures. Beyond this, we have seen that there are no recognizable patterns, which suggests that relative error estimation does not improve the situation. The empirical results provide strong arguments that, without additional discrete techniques, the problem cannot be solved completely.

Algorithm. In our algorithm, we first generated a rough approximation of the topology with a conventional approach. This source of a discrete structure was good enough to be used by a discrete algorithm that applied logical inference rules to correct a lot of errors in the topology. The output of the algorithm is in many cases satisfying.

Checker. However, the verification of the output is crucial, because of the dirtiness of the input and the subjectivity of the output. The "check by eye" facilitates this task. A check of each neighbourhood would be as time consuming as the generation of the output by hand and is therefore infeasible.

Final remark. In analysing this problem, a difficulty becomes apparent that seems to be inherent to various practical problems. The standard strategy of analysing a problem, that is raising it to an abstract, mathematical model and solving the abstract problem efficiently, does not work properly here. One has to "stay in the mud."

Bibliography

[1] Gill Barequet and Micha Sharir. Filling gaps in the boundary of a polyhedron. *Computer Aided Geometric Design*, 12:207-229, 1995.

[2] Jan Helge Bøhn and Michael J. Wozny. A Topology-Based Approach for Shell-Closure. *Geometric Modeling for Product Realization*, IFIP Transactions B-8 p. 297-319, 1993.

[3] Georges M. Fadel and Chuck Kirschman. Accuracy issues in CAD to RP translations. In *Internet Conference on Rapid Product Development*. MCB University Press, 1995. `http://www.mcb.co.uk/services/conferen/dec95/rapidpd/fadel/fadel.htm`.

[4] Günter Krause. Interactive finite element preprocessing with ISAGEN. *Finite Element News*, 15, 1991.

[5] Rolf H. Möhring, Matthias Müller-Hannemann, and Karsten Weihe. *Mesh refinement via bidirected flows; modelling, complexity, and computational results*. Journal of the ACM 44(3), p. 395-426, 1997.

[6] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1994.

[7] Stephen J. Rock and Michael J. Wozny. Generating topological information from a 'bucket of facets'. In H.L. Marcus et al., editors, *Solid Freeform Fabrication Symposium Proceedings*, pages 251-259, 1992.

[8] Xuejun Sheng and Ingo R. Meier. *Generating Topological Structures for Surface Models*. IEEE Computer Graphics and Applications Vol. 15, No. 6. 1995.

[9] Su Bu-qing and Liu Ding-yuan. *Computational Geometry – Curve and Surface Modeling*. Academic Press, Inc., 1989.

[10] Karsten Weihe and Thomas Willhalm. Reconstructing the Topology of a CAD model - a discrete approach. *Proceedings of the 5th European Symposium on Algorithms, ESA 1997*, Springer Lecture Notes in Computer Science 1284, p. 500-513.
(Journal version to appear in *Algorithmica*)

[11] Karsten Weihe and Thomas Willhalm. Why CAD Data Repair Requires Discrete Algorithmic Techniques. 2nd Workshop on Algorithm Engineering WAE 1998, Saarbrücken, Germany. `http://www.mpi-sb.mpg.de/~\/wae98/PROCEEDINGS/`

Appendix A

Diagrams about Relative Error Estimation

This appendix contains the diagrams as described in Section 4.1. The functions are scaled to fit in the diagrams. The units of the threshold value and the respective properties depend on the units of the data files. They are quite arbitrary, because the designer of a workpiece is free to choose the scaling of the workpiece.

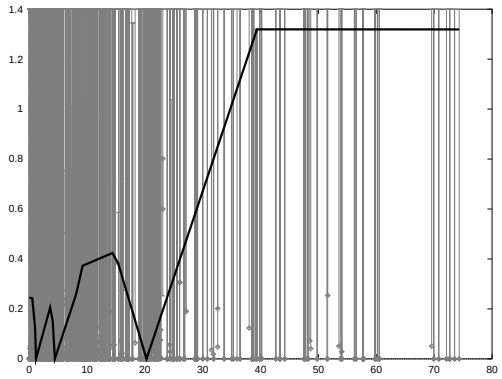

Figure A.1: Approximating function for a possible threshold value depending on the length of edges for workpiece 3,

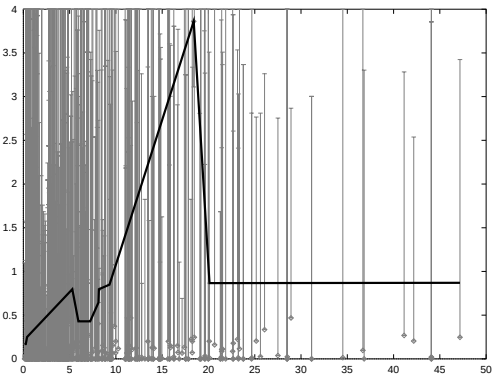

Figure A.2: depending on the length of edges for workpiece 12,

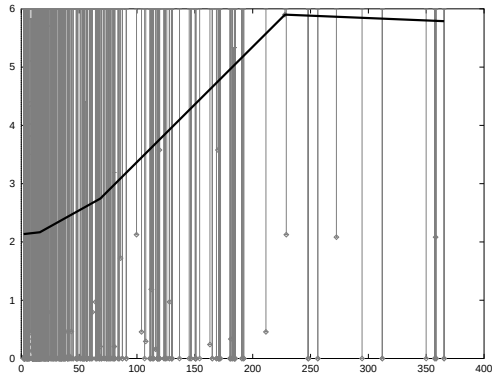

Figure A.3: depending on the length of edges for workpiece 8, and

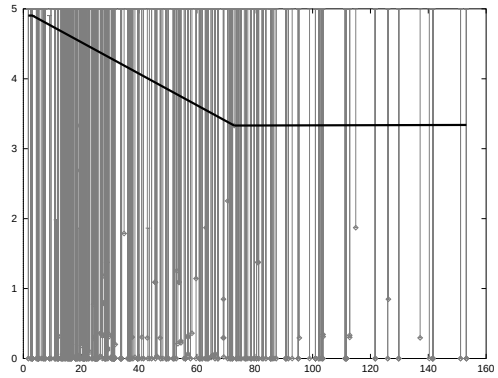

Figure A.4: depending on the length of edges for workpiece 13.

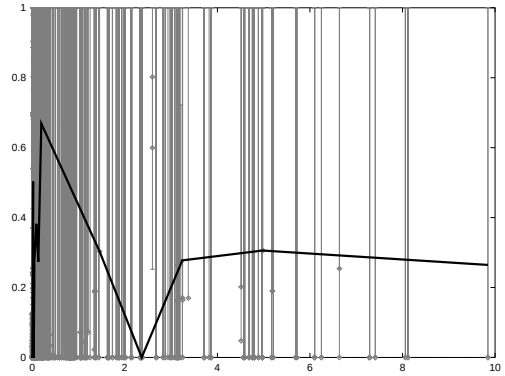

Figure A.5: Approximating function for a possible threshold value depending on the curvature of edges for workpiece 3,

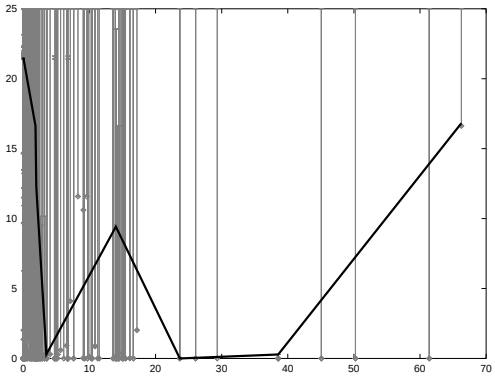

Figure A.6: depending on the curvature of edges for workpiece 6,

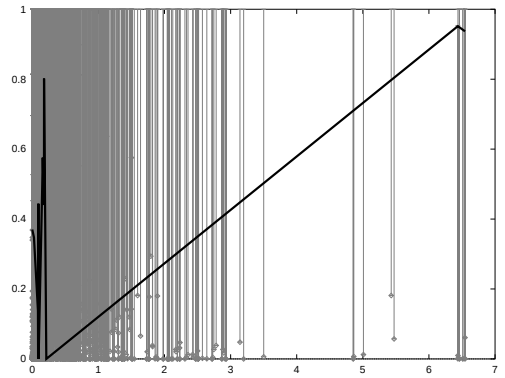

Figure A.7: depending on the curvature of edges for workpiece 7, and

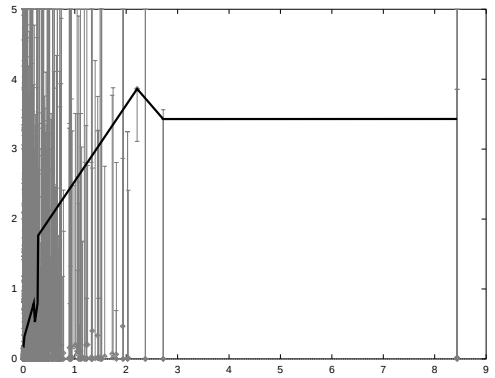

Figure A.8: depending on the curvature of edges for workpiece 12.

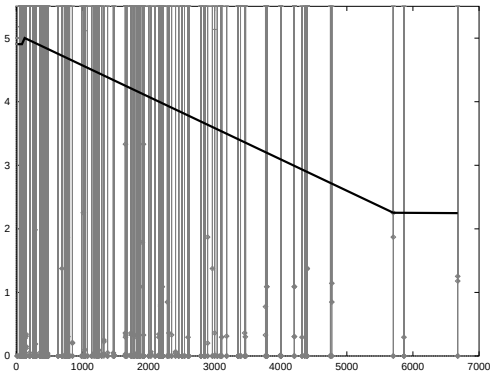

Figure A.9: Approximating function for a possible depending on the area of mesh elements for workpiece 13,

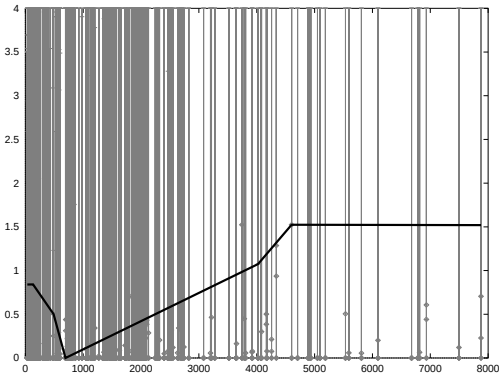

Figure A.10: depending on the area of mesh elements for workpiece 14,

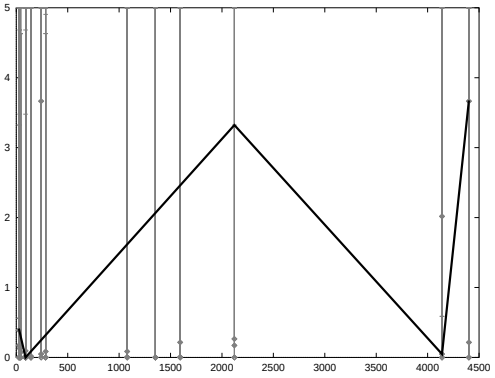

Figure A.11: depending on the area of mesh elements for workpiece 1, and

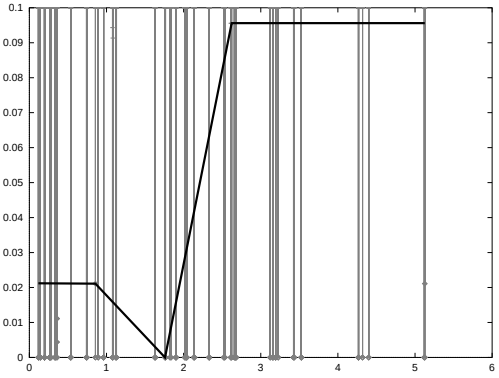

Figure A.12: depending on the area of mesh elements for workpiece 16.

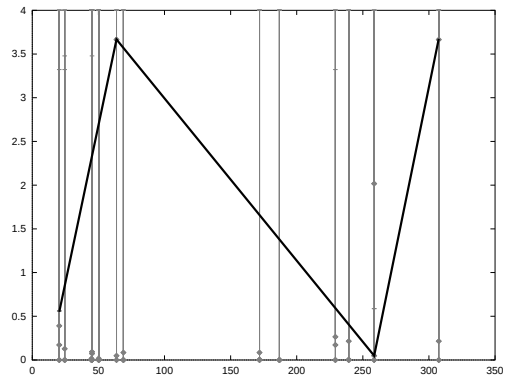

Figure A.13: Approximating function for a possible depending on the perimeter of mesh elements for workpiece 1,

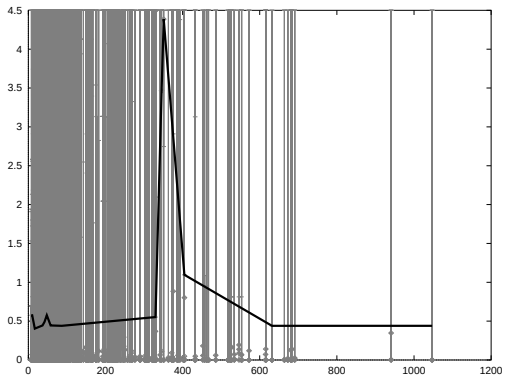

Figure A.14: depending on the perimeter of mesh elements for workpiece 7,

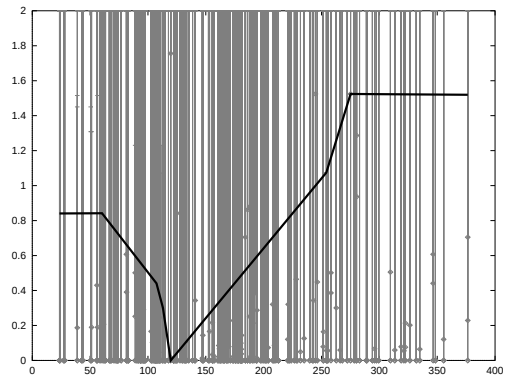

Figure A.15: depending on the perimeter of mesh elements for workpiece 14, and

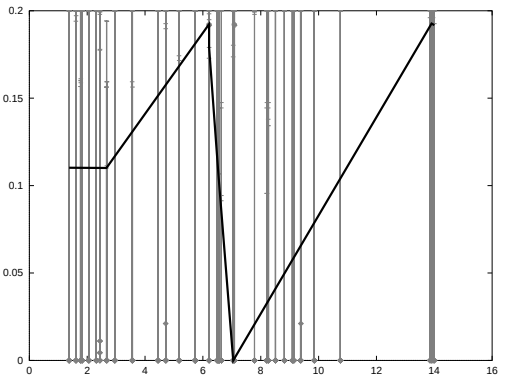

Figure A.16: depending on the perimeter of mesh elements for workpiece 16.

Appendix B

Images of Workpieces

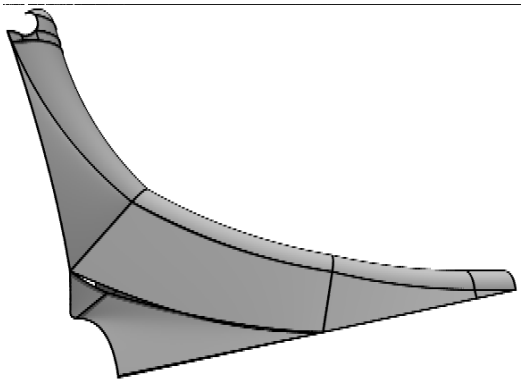

Figure B.1: Workpiece 1

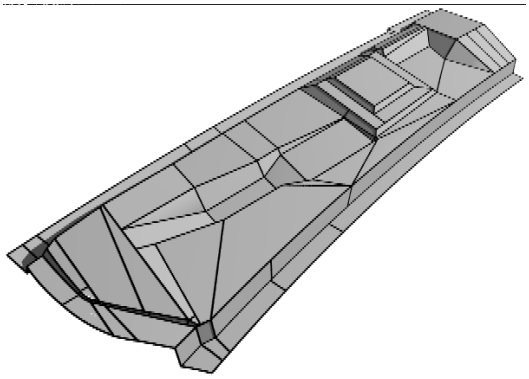

Figure B.2: Workpiece 2

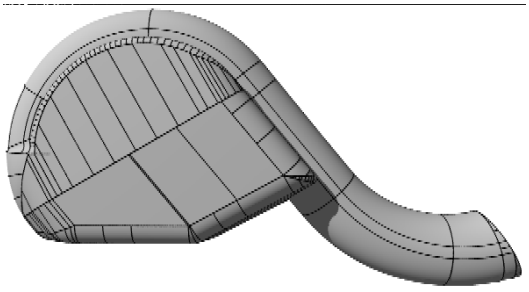

Figure B.3: Workpiece 3

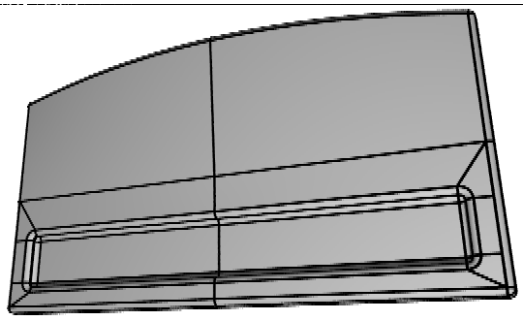

Figure B.4: Workpiece 4

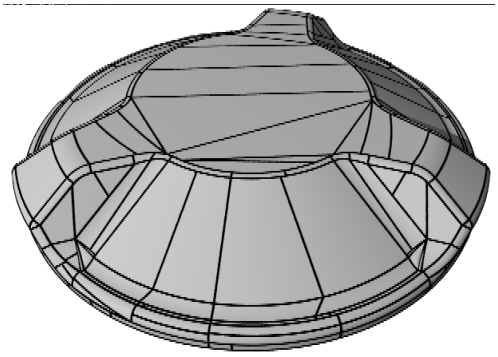

Figure B.5: Workpiece 5

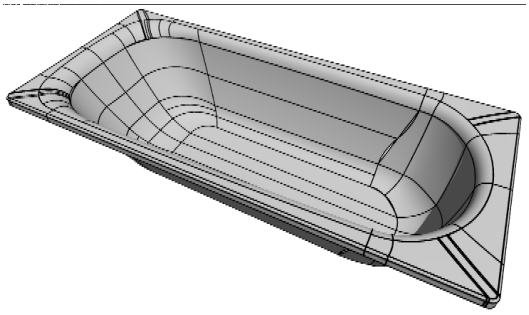

Figure B.6: Workpiece 6

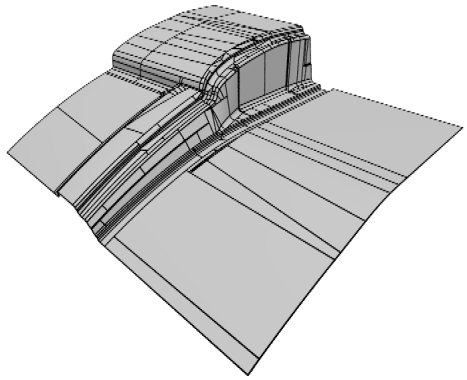

Figure B.7: Workpiece 7

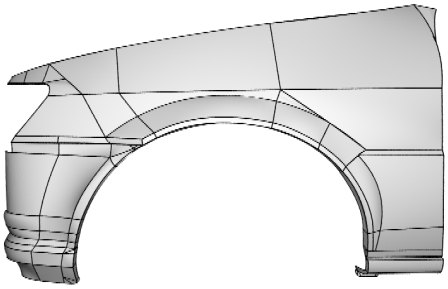

Figure B.8: Workpiece 8

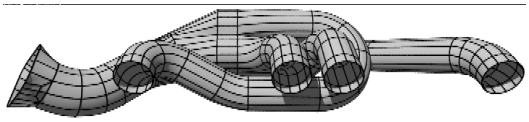

Figure B.9: Workpiece 9

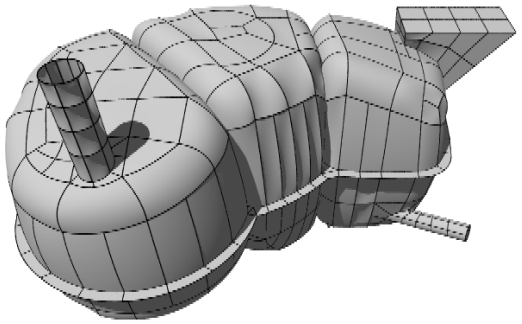

Figure B.10: Workpiece 10

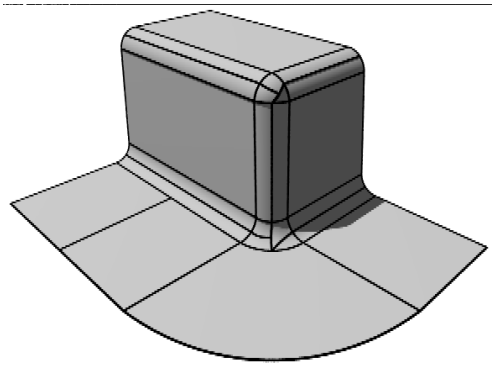

Figure B.11: Workpiece 11

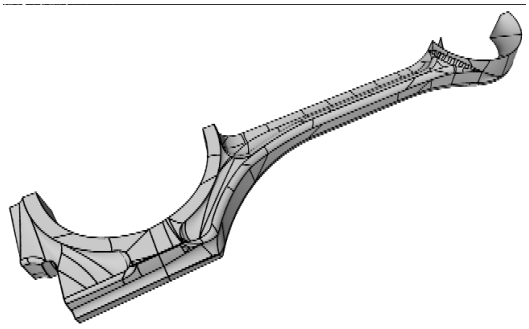

Figure B.12: Workpiece 12

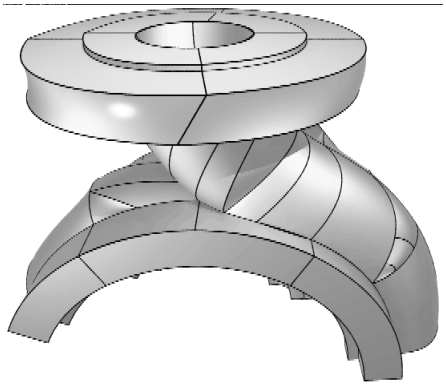

Figure B.13: Workpiece 13

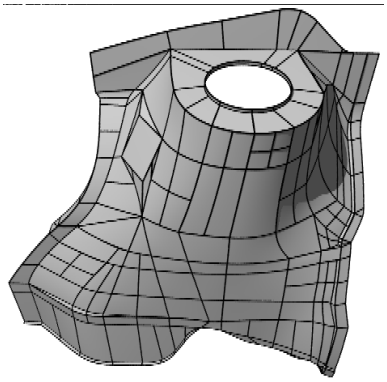

Figure B.14: Workpiece 14

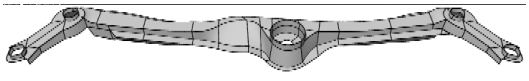

Figure B.15: Workpiece 15

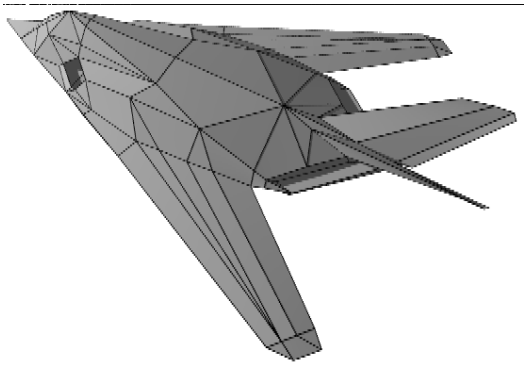

Figure B.16: Workpiece 16

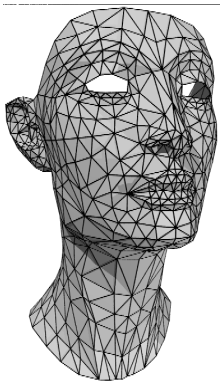

Figure B.17: Workpiece 17

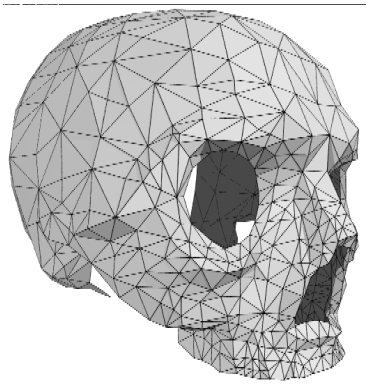

Figure B.18: Workpiece 18

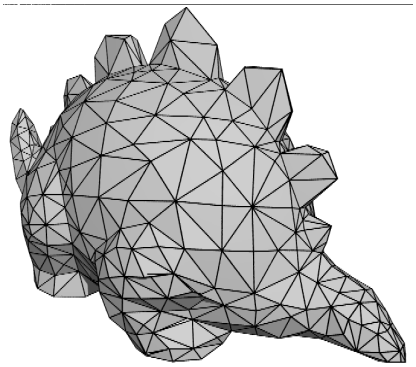

Figure B.19: Workpiece 19

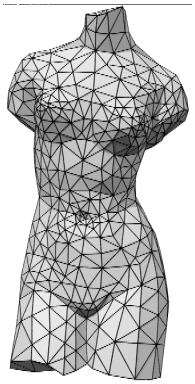

Figure B.20: Workpiece 20