FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

3D Mesh Unfolding

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Thorsten Korpitsch
Matrikelnummer 01529243

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: PhD Hsiang-Yun Wu

Wien, 1. März 2019

Thorsten Korpitsch	Hsiang-Yun Wu

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

3D Mesh Unfolding

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Thorsten Korpitsch
Registration Number 01529243

to the Faculty of Informatics
at the TU Wien
Advisor: PhD Hsiang-Yun Wu

Vienna, 1st March, 2019

Thorsten Korpitsch

Hsiang-Yun Wu

Erklärung zur Verfassung der Arbeit

Thorsten Korpitsch

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

Wien, 1. März 2019

Thorsten Korpitsch

Danksagung

Ich möchte mich bei meiner Beraterin Ph.D. Hsiang-Yun Wu bedanken, dass sie mich in das Thema 3D-Netzfaltung eingeführt hat und mir unermüdlich mit Rat, Tat und Anleitung zur Seite stand. Insbesondere möchte ich ihr auch dafür danken, dass sie mir 3D-Modelle zur Verfügung gestellt hat, um meinen in dieser Arbeit beschriebenen Ansatz zu evaluieren.

Außerdem möchte ich mich bei meinen Eltern, Ingrid und Horst Korpitsch, sowie bei meiner Großmutter Erna Kaiper und meiner besseren Hälfte Valentina Bone bedanken, dass sie mich auch in schwierigen Zeiten immer unterstützt und ermutigt haben. Ohne ihre ewige Geduld, Unterstützung und Motivation wäre diese Arbeit nicht möglich gewesen.

Ich möchte auch Nils Voß MSc für das Korrekturlesen dieser Arbeit danken.

Acknowledgements

I want to thank my advisor, PhD Hsiang-Yun Wu for introducing me to the topic of 3D-Mesh Unfolding and tirelessly providing me with advice, action and guidance. I especially also want to thank her for providing me with prepared 3D-Models to evaluate my approach described in this thesis.

Furthermore, I want to thank my parents, Ingrid and Horst Korpitsch, as well as my grandmother Erna Kaiper and my better half Valentina Bone for always supporting and encouraging me even in hard times. Without their everlasting patience, support and motivation, this thesis would not have been possible.

I also want to thank Nils Voß MSc for proofreading this thesis.

Kurzfassung

3D Mesh Unfolding ist der Prozess der Transformation des 3D Mesh in eine planare Fläche. Ein Einsatzgebiet für diesen Prozess ist die Kunst des Papierhandwerks. Während des Verfahrens können zwei signifikante Probleme auftreten. Erstens können sich aufgefaltete Flächen überlappen, was es unmöglich macht, einen einzelnen Ausschnitt zu erstellen. Zweitens dürfen die Flächen beim Entfalten nicht verzerrt werden, denn das würde die Rekonstruktion des Originalmodells unmöglich machen. Diese Arbeit konzentriert sich auf das Hinzufügen von Klebstellen an den Begrenzungskanten, um die Rekonstruktion für die Benutzer zu erleichtern, da sie den Benutzern Raum zum Auftragen von Klebstoff bietet. Ein minimaler Spannbaum des Dualgraphen des ursprünglichen Netzes wird berechnet, um das Netz und die berechneten Klebstellen zu entfalten. Mit dem Hinzufügen von Klebstellen müssen nicht nur Flächenüberlappungen behandelt werden, sondern auch Überlappungen zwischen Flächen und Klebstellen und Klebstellen untereinander. Ein simulierter Glühprozess optimiert die Entfaltung und löst Überlappungen auf.

Abstract

3D Mesh Unfolding is the process of transforming a 3D mesh into a 2D planar patch. A field of use for this process is the art of papercraft. During the procedure, two significant problems can arise. Firstly unfolded faces may overlap each other, which makes it impossible to create a single cutout. Secondly, during unfolding, the faces must not be distorted, because that would defeat the purpose of reconstructing the original model. This thesis focuses on the addition of glue tags on boundary edges to make reconstruction easier for users, as it gives users space to apply glue. A spanning tree of the dual graph of the original mesh is calculated to unfold the mesh and the calculated glue tags. With the addition of glue tags not only face overlaps need to be treated but also overlaps between faces and glue tags and glue tags with each other. A simulated annealing process optimizes the unfolding and resolves any overlaps.

Contents

CHAPTER 1

Introduction

1.1 Background

Papercraft is a widely popular art of creating two or three-dimensional objects from cardboard or paper. The models that are created range from simple ones, like paper aeroplanes, to elaborate models of buildings or districts for planning. Further, it can be used in combination with self-folding materials to form structures after printing the planar patch. In order to build papercraft models a 3D mesh representing the object needs to be unfolded into a 2D mesh, then this mesh can for example be printed onto paper and then reconstructed into the 3D model. It is a complex task due to two kinds of conflicts that appear in the unfolding process. Distortion of the model, as well as faces overlapping each other, are problems that occur when unfolding a 3D mesh whereas both should be avoided to allow authentic reconstruction.

1.2 Motivation

As the reconstruction of a model can be quite tricky, even with indicators helping with glueing the right faces together, like the solution presented by Takahashi et al. [TWS+11], glueing the faces together if they are tiny is still a hard task. To make reconstruction easier this paper introduces *glue tags*. These glue tags add small faces on edges that are cut and give user space to apply glue to. Adding glue tags, which can improve the reconstruction experience, has not been well explored as most of the previous work focused on finding ways to get highly qualitative unfoldings. Attaching glue tags makes the problem more difficult as the solution space for a solution without overlaps shrinks, whereas the search space of possible unfoldings and possible glue tag positions increases.

1.3 Goal

The goal of this thesis is to explore the addition of glue tags to the cut Edges. This thesis proposes the calculation of glue tags in advance for the 3D-Model before the unfolding process starts. During the unfolding glue tags are treated as part of the original mesh. Hence there is no need for additional calculations. This thesis proposes a simple spanning tree approach to find unfoldings of the mesh. A simulated annealing process is optimizing the search for an overlap free unfolding by finding an optimal global layout. Steps of the process are visualized and allow users to interact with both the 3D-Model and the final planar patch. At last, the performance and limitations of the proposed approach are evaluated.

1.4 Results

The suggested approach generates results in a limited amount of time. Experiments strongly suggest that the time frame for finding unfoldings increases as meshes increase their number of faces, but also the layout of faces influences the time to find unfoldings. Meshes with less than 200 triangles are unfolded consistently without any overlaps remaining, whereas meshes with up to 400 triangles can be unfolded with an increased timeframe and meshes over 700 faces cannot be resolved. Another limiting factor is the size of glue tags, which makes finding overlap free unfoldings less likely as glue tags increase in size. To counteract the increase in time needed for a greater amount of faces, the glue tag size can be adjusted.

1.5 Structure

First chapter 2 provides an overview of previous findings related to 3D Mesh Unfolding and highlights differences to the proposed approach in this thesis. It also provides the theoretical background and related work for simulated annealing, that optimizes the search for unfoldings. Chapter 3 defines the concepts of dual graphs and minimum spanning trees and further describes the data structure for this approach. The next chapter, 4, gives an overview of the processing pipeline, that calculates an unfolding and describes necessary steps in detail. The next chapter, 5, brings insight into the implementation of the previously explained approach and focuses on the simulated annealing process. Chapter 6 shows the results of the implemented approach and discusses and evaluates its performance and limitations. The last chapter 7 summarises the findings and gives an outlook on future work.

CHAPTER 2

Related Work

2.1 3D-Mesh Unfolding

According to Takahashi et al. [TWS+11] unfolding 3D-Meshes, more concrete polyhedron models, into a single patch is still a well-known open problem. They have proposed a heuristic approach using a genetic-based algorithm to find distortion free unfoldings. The key concept is to use topological surgery to construct models by stitching together boundary edges of the unfolded mesh.

Straub et al. [SP11] explored the unfolding and calculating gluetags to an unfolded mesh, in which they also explore the removal of overlaps by introducing new subdivisions to the mesh. As in the previously mentioned paper, a heuristic approach is used to calculate possible unfoldings. They use a greedy algorithm to optimize the cut out and to resolve overlaps. Gluetags that have been calculated to an unfolding are optimized after an unfolding is found.

A different approach is proposed by Jun Mitani and Hirosama Suzuki [MS04]. In their paper, they describe producing unfoldings by using strip-based approximation. They propose segmenting meshes into parts of easily reconstructible segments, which can be done with feature extraction. They add internal cut lines whenever a feature, like a dent in the mesh, is detected on a triangle strip, to make it reconstructible. Due to this simplification they produce a rather large error compared to other simplification methods. Their solution mostly focused on large mesh models where they merged regions between 60 and 250 triangles.

Chang et al. [XKKL16] have focused their work to unfold orthogonal polyhedra. They propose algorithms to unfold 1-layer orthogonal polyhedra by introducing additional cuts depending on their genus. Their main concept is to find a subset of faces, that traverses the surface of the polyhedron, to unfold it along its line.

Theoretical approaches for unfolding meshes have been intensively explored [She75], while other papers focus on different kind of meshes, for example a lot of papers focus on the unfolding of orthogonal polyhedra [XKKL16][DFO07][DDF14]. Yet the focus on the reconstruction experience has been neglected. This thesis focuses on exploring the addition of glue tags to triangulated polyhedral meshes. They should help users in the reconstruction effort to glue cut edges together, as well as give a better indication of which edges should be glued together. Therefore a simpler approach to calculating unfoldings using a minimum spanning tree approached, optimized by simulated annealing was selected.

2.2 Simulated Annealing

Simulated annealing is a well-known optimization process [KGJV83] that is widely applicable in problems found in computer science [GFR94] [DA91] [BM95] and other scientific fields [PBARCC90] [SDJ95]. This process tries to emulate the annealing of metal that is being processed.

The main goal is to find an optimal value of a function, which is called the cost function, which has many independent variables. A typical example that simulated annealing is applied to is the travelling salesman problem [MGPO89], where the most effective route between different cities needs to be found.

Simulated annealing is an iterative process that starts with a system and its configuration P. For this configuration, the cost function calculates a value, which is called the energy of the configuration E. In each iteration, the configuration P is rearranged to a configuration P', for which the cost function calculates the value E'. Let $E' \leq E$ be true, then the new configuration P' replaces P as the base configuration, from which new configurations are arranged. This iteration repeats until a temperature T reaches a defined minimum temperature T_{min}, as T cools down in each iteration. When T reaches T_{min} the cost E of configuration P is minimized.

The process has to be extended with another step to counterackt the problem of deadlocking, which happens when a configuration P reaches a local minimum, instead of the global one. To avoid such situations, let $\Delta E = E - E'$ be the difference between the energy of the old configuration and the energy of the new configuration. If $\Delta E \leq 0$ holds true the new configuration is accepted, otherwise if $\Delta E > 0$ the new configuration is treated probabilistic, where the chance of accepting it is $(\Delta E) = exp(-\Delta E/k_B T)$, where k_B is the Boltzmann constant. Now a uniformly distributed random number R within the range of $(0, 1)$ implements the random part of the algorithm. If $R < P(\Delta E)$ then the new configuration is accepted. Otherwise, it discards the new configuration. As T decreases over time, it gets more unlikely to accept worse configurations as time passes on and the algorithm nears its end.

To summarize simulated annealing, four essential ingredients are necessary. First, a concise description of the configuration of a system is required. Second, a generator,

which generates random moves, that changes the configuration of a system is needed. A quantitive function that evaluates the trade-offs for each iteration needs to be defined. Last but not least, the system can only evolve with an annealing schedule of temperatures and length of times. This evolving process can be stopped with a cooling-rate, which tries to emulate the cooling down of metal, because metal can only be formed while it is hot.

CHAPTER 3

Definitions

3.1 Data

The original data has to meet the following requirements to be applicable.

- Data is in the Object File Format (.off)
- Data is triangulated
- Doubled Vertices are removed
- Mesh has no holes
- All faces are connected

The data is read from a file and saved into the CGAL data structure Polyhedron_3 [The19]. A Polyhedron_3 object consists of vertices, edges and facets and an incidence relation on them, as shown in Figure 3.1. An halfedge and an opposite halfedge, that points into the opposite direction, represent each edge of the mesh. These halfedges consist of vertex pairs that can be accessed. The opposite halfedge links each face to its neighbouring face, also the halfedges are also linked together, therefore enabling iterating through all halfedges of a face. With this data structure, all information is available to apply the approach suggested in this thesis.

3.2 Dual Graph

Let G_M be the graph representation of a mesh, which is defined by its vertices V and undirected edges E between the vertices. $G_{M_d} = (V_d, E_d)$ is called the dual graph of a graph $G_{M_d} = (V, E)$, which can be obtained by calculating a dual vertex V_d in each

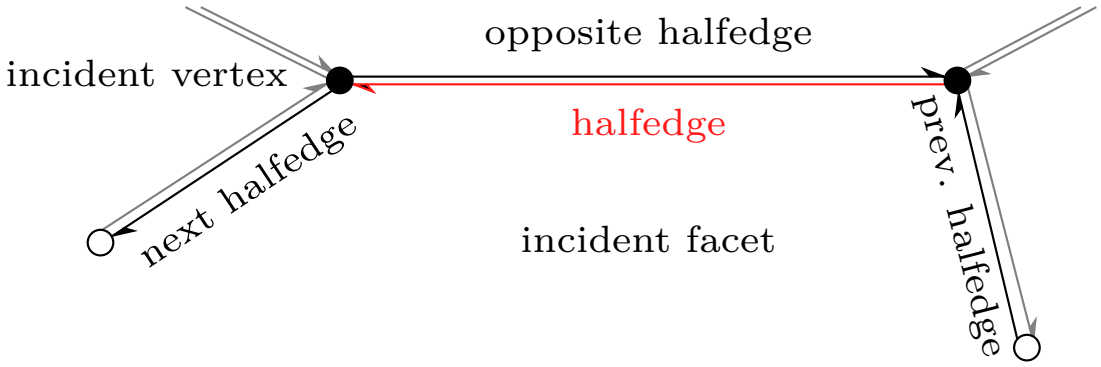


Figure 3.1: CGAL Polyhedron_3 datastructure visualized. Adapted from the CGAL user manual [The19].

enclosed facet and an dual-edge E_d for every two facets separated by an edge in E [GY04], as seen in Figure 3.2. The dual graph can then be used to find an unfolding, as a dual-edge connects each neighbouring facets. These dual-edges can either represent an edge that is cut or an edge that is used for bending. Therefore the dual graph contains all edges that connects the faces in an unfolding, but also all edges that are cut. A graph has only one dual graph and as the calculation of it can be done very efficient, which makes it a good option to use in 3D mesh unfolding for finding an unfolding.

3.3 Minimum Spanning Tree

Let $G = (V, E)$ be a connected undirected graph with $|V| = n$ vertices and $|E| = m$ edges. Given a value $c(v, w)$ for each edge $(v, w) \in E$, a spanning tree $T = (V, E'), E' \subseteq E$ such that $\sum_{\{v,w\} \in E'} c(v, w)$ is minimal [CT76]. A minimum spanning tree is therefore acyclic by definition and can be used to find an unfolding, as an unfolding must not contain cycles. Many simple algorithms for computing minimum spanning trees are available [Kru56] [AMOT90].

If the minimum spanning tree is calculated from the dual graph the edges that are part of the minimum spanning tree are the edges that are bent when reconstructing. All other edges not in the minimum spanning tree arre edges that are cut. Therefore a minimum spanning tree is due to it's simple calculation a good combination with the dual graph to calculate possible unfoldings.

As a weighted graph, where no edges have the same weight, have exactly one minimum spanning tree. This is a problem, since the dual graph doesn't inherently have weights to begin with. Weights need to be assigned to each edge of the dual graph. But the weights cannot be constant values, as in the traveling salesman problem, because not all minimum spanning trees will yield an overlap without any overlaps. As initial weights the length of the edges of the original graph, that connect the faces with each other, can

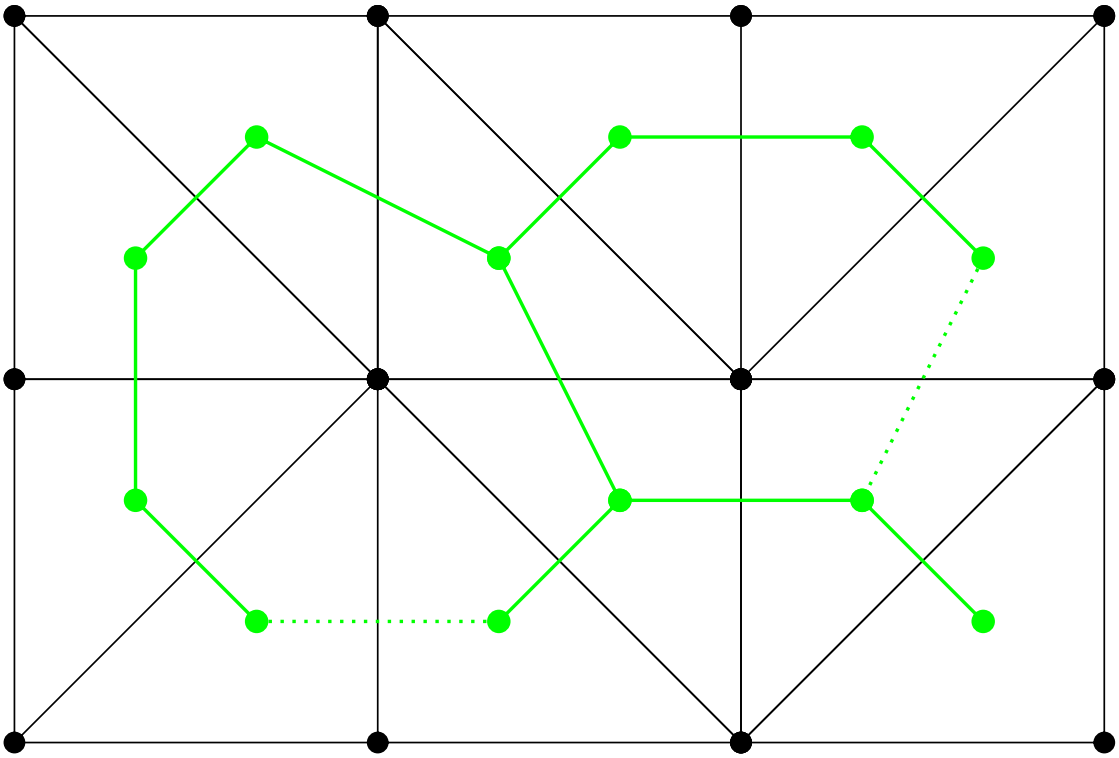


Figure 3.2: (black) Shows a graph with undirected edges. (green solid and dotted) Shows the graphs complete dual graph. (green solid) Shows a minimum spanning tree of the dual graph.

be used as a weight.

CHAPTER 4

Methodology

4.1 Overview

After loading a mesh, its dual graph is calculated as a first step, as seen in Figure 4.1, as it is necessary for all later calculations. After that for each edge, two glue tags are calculated, each targeting one facet of the edge and having its source on the other facet, see Figure 4.2. Then the simulated annealing process starts. Each iteration calculates and unfolds a new minimum spanning tree. Afterwards overlaps are calculated, if overlaps are found a new minimum spanning tree is calculated. If no overlaps are found, the process ends, as an overlap-free solution has been found. In the following subchapters, each Step is described in more detail, whereas the simulated annealing process is explained in more

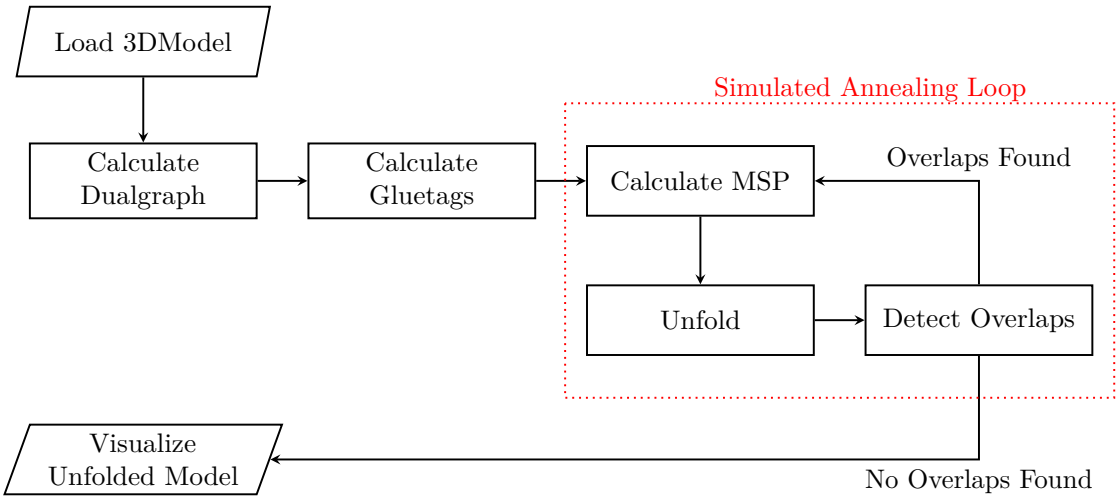


Figure 4.1: Overview of the 3D Mesh-Unfolding Process.

detail in Chapter 5.

4.2 Calculate Dualgraph

As the dual graph is calculated from the original mesh, which does not change during the unfolding process, the dual graph is calculated at the beginning once, as shown in Figure 4.1. The neighbourhood relation of the facets can be derived from the half edges connecting the mesh vertices.

The dual graph is calculated by iterating through all facets of the mesh. For each facet, we need to iterate through the half edges, where the opposite half-edges are used to identify the neighbouring face. These two facets are saved as an edge and can be initialized with a weight, for example, the length of the original half-edge separating the faces or a random value.

4.3 Calculate Gluetag

The second step in the pipeline shown in Figure 4.1 is to calculate glue tags. For each edge of the dual graph, a glue tag is calculated for both facets connected by this edge. The vertices of an edge are the base of a glue tag. As glue tags, an example shown in Figure 4.2, can vary in shape and size, this thesis proposes glue tags in the shape of a trapezoid as it brings a few advantages.

Two triangles define a trapezoid. Therefore algorithms that are applied to triangles of the mesh can be applied to the glue tags without altering them. As the top of the trapezoid is smaller than the base, glue tags that are placed next to each other are less likely to overlap compared to rectangular glue tags. The algorithm calculates the height of the glue tag depending on the targeted facet so that it takes up a maximum of 20 per cent of the targeted facets space.

Glue tags are calculated once after the dual graph was calculated, as the edges they are linked to do not change. The algorithm chooses the glue tags based on the calculated minimum spanning tree.

4.4 Calculate Minimum Spanning Tree

In the first step, as shown in Figure 4.1, the algorithm calculates a minimum spanning tree from the dual graph. This is done using Kruskal's Algorithm [Kru56] to find the shortest spanning subtree.

For this, the edges are sorted by their weight in ascending order. Then the algorithm iterates through all edges and adds each edge to an adjacency list. This list is used to check if the recently added edge causes the graph to be cyclic. If this is the case, the last edge is removed and added to a list containing the cut edges. Otherwise, the edge

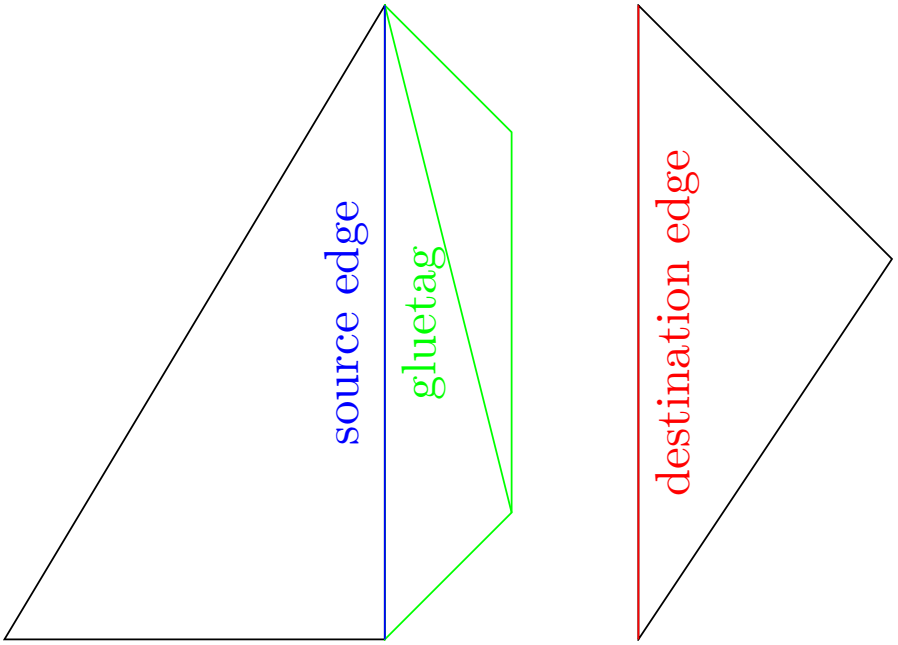


Figure 4.2: Exemplary trapezoid gluetag (green) attached to its source triangle edge (blue) and the triangle it will be glued to (red).

is added to a list containing the bend edges. Furthermore, for each iteration, we save the information which face is discovered, so we can assure that the graph is a connected graph besides being acyclic, which is necessary. As a result, the adjacency list defines a minimum spanning tree.

To be able to calculate different minimum spanning trees the weights of an edge is changed on each iteration, which is part of the simulated annealing process. Fur further details see chapter 5.

4.5 Unfolding

Following the calculation of the minimum spanning tree, as shown in Figure 4.1, an unfolding of the mesh is calculated. The adjacency list of the last step can be used to determine the order of the faces for unfolding.

Let $T = (A, B, C)$ be a triangle defined by three vertices $A(x_A, y_A, z_A)$, $B(x_B, y_B, z_B)$ and $C(x_C, y_C, z_C)$, where x_N, y_N and z_N, with $N \in \{A, B, C\}$, are the coordinates of each vertex. Let $T_p = (a, b, c)$ be the planar representation of the given triangle, defined by $a(x_a, y_a)$, $b(x_b, y_b)$ and $c(x_c, y_c)$, where again x_n, y_n, with $n \in \{a, b, c\}$, are the defining coordinates of each vertex.

The first face is treated as a particular case, as the later triangles depend on the position of the vertices of the first triangle. Let A be the vertices of the first triangle, and it is set

to $(0, 0)$ disregarding the position of A as it is not relevant. The algorithm calculates b by calculating the distance $\overline{AB}$ as it is equal to $\overline{ab}$, which can be calculated using formula 4.1.

$$\overline{AB} = \overline{ab} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2} \tag{4.1}$$

Now b can be set to $(\overline{AB}, 0)$, with y being set to 0 as the original orientation of the triangle does not need to be retained. The last vertices c is again a particular case, as two positions are possible, which will be necessary for all vertices except the first. Two solutions are possible for c, as it can be placed on either side of the vector $(a - b)$.

$$\begin{aligned} s &= \frac{\|(B - A) \times (C - A)\|}{(\overline{AB})^2} \\ d &= \frac{(B - A) \cdot (C - A)}{(\overline{AB})^2} \\ c_x &= a_x + d(b_x - a_x) - s(b_y - a_y) \\ c_y &= a_y + d(b_y - a_y) + s(b_x - a_x) \end{aligned} \tag{4.2}$$

The first candidate solution for $c_1 = (c_x, c_y)$ is defined by the formulas 4.2. The second candidate solution for c_2 can be calculated by the formulas 4.3.

$$\begin{aligned} c_x &= a_x + d(b_x - a_x) + s(b_y - a_y) \\ c_y &= a_y + d(b_y - a_y) - s(b_x - a_x) \end{aligned} \tag{4.3}$$

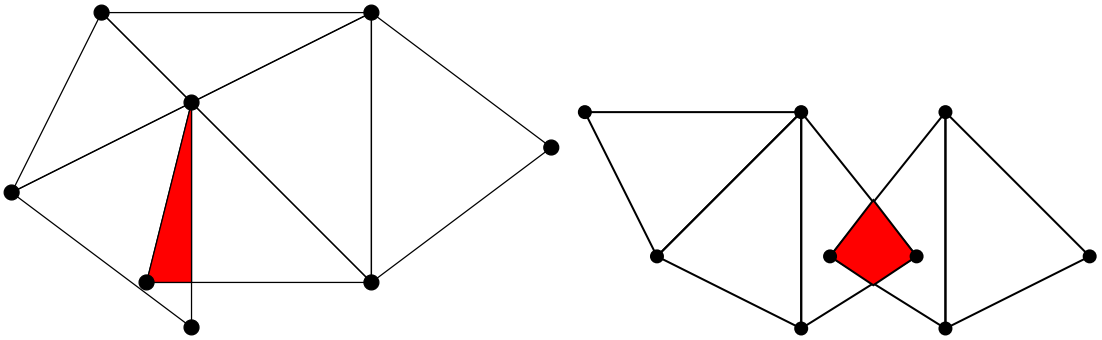
For the first triangle, any of the two solutions can be chosen. For all other triangles a check needs to be performed, so that c is not on the same side as it is for the previous triangle.

Now let $c_{prev} = (c_{x_{prev}}, c_{y_{prev}})$ be the vertex of the previous triangle that it does not share with the current triangle. To calculate the side on which the vertex c has to be put can be determined by the formula 4.4.

$$f = (c_x - a_x) * (b_y - a_y) - (c_y - a_y) * (b_x - a_x) \tag{4.4}$$

In theory these formulas can yield three different results $f < 0$ for one side, $f > 0$ for the other side and $f = 0$ if it is on the line. But as we do not have any degenerate triangles in our mesh there are two cases. f needs to be calculated for c_{prev} and for the two solutions of the previous formula c_1 and c_2. Now we are left with two possibilities, if $f_{prev} < 0$ and $f_{c_1} > 0$ or if $f_{prev} > 0$ and $f_{c_1} < 0$ then we set $c = c_1$, otherwise $c = c_2$.

The unfolding of glue tags works in an analogue way, as the glue tag consists of two triangles, therefore it is left to the reader as an exercise.



(a) Two triangles overrlap each other, resulting in a triangle describing the red overlap area.

(b) Two triangles overlap each other, resulting in a polygon describing the red overlap area.

Figure 4.3: Overlaps of triangles resulting in different overlapping areas.

4.6 Overlap Detection

The last step of the simulated annealing loop, as shown in Figure 4.1, is to determine the quality of the unfolding. As a measurement for the quality, merely the sum of the overlapping areas is used. Overlaps, as seen in Figure 4.3, can happen between triangles, triangles and glue tags or between glue tags. The detection is a two-step process. First, only the triangles are checked if they overlap each other. In a second step, if no triangles overlap each other, the glue tags are checked if they overlap each other or a triangle. The algorithm checks if any of their lines intersect with each other to find out if triangles overlap each other.

Let $T = (A, B, C)$ and $S = (P, Q, R)$ be two triangles that might be overlapping each other, defined by their points A, B, C, P, Q, R. Therefore we check the lines $\overline{AB}$ and $\overline{PQ}$ if they intersect with each other. For two line segments we can define a general case for overlaps and a special case. In general the lines intersect only if the points (A, B, P) and (A, B, Q) have differrent orientations and (P, Q, A) and (P, Q, B) have different orientations.

For the special case, the lines intersect, if all four point-triplets are collinear and the x-projections of (A, B) and (P, Q) intersect and the y-projections of (A, B) and (P, Q) intersect as well.

The proposed solution calculates for every pair of lines of the two triangles if they intersect. If at least one pair of lines intersect each other, it means that the triangles intersect. The overlap detection does not need to be done with the child of another triangle as they share an edge and therefore cannot overlap, as the child is unfolded onto the other side of the shared edge, as explained in section 4.5.

After detecting that an overlap occurs, the algorithm calculates the area of the polygon, which describes the overlap, to determine the energy of the current configuration, the overlapping area is calculated using the Sutherland-Hodgman Clipping algorithm [SH74].

Algorithm 4.1: Sutherland-Hodgman pseudo algorithm.

```
Data: Clipping Polygon, Subject Polygon
Result: List of vertices of the intersection polygon
List output = Subject Polygon Vertices;
while Edge clipedge in Clipping Polygon do
    List input = output;
    clear output;
    Point S = input.last();
    while Point E in inputlist do
        if E inside clipedge then
            if S not inside clipedge then
                output.add(ComputeIntersection(S,E,clipedge));
            end
            add E to output;
        else
            if S inside clipedge then
                output.add(ComputeIntersection(S,E,clipedge));
            end
            S = E;
        end
    end
end
return output;
```

This algorithm finds the vertices of the intersection polygon created by the two triangles. The overlap can be described as a polygon, it can be either be defined at least as a triangle, see Figure 4.3a, or a polygon with a degree, see Figure 4.3b, depending on how the triangles overlap each otherr. Algorithm 4.1 shows the pseudocode for the Sutherland-Hodgman Clipping algorithm, where the input is two polygons, in our case, the two triangles that intersect. The vertices list resulting from this algorithm can be used to calculate the area using the shoelace formula [ŠVV17], see formula 4.5 where x_i and y_i are the coordinates of the i-th point in P. It is a mathematical algorithm to calculate the area of a simple polygon, that is described by ordered points P in a plane.

$$Area = \frac{1}{2}\left|\sum_{i=0}^{P-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right| \tag{4.5}$$

CHAPTER 5

Implementation

5.1 Specifications

The prototype is implemented on a Laptop with an Intel Core i7 CPU (3.3 GHz, 4MB Cache) and 8GB RAM. The source code is written in C++17 and the OpenGL Library library is used for the visualization of the algorithms step as well as the results. The CGAL library is used to read in off-Files as well as the underlying data structure. The Graphical User Interface (GUI) is developed using the Qt Library. CMake is used to manage the build process, and it compiles on an Ubuntu 18.04.02 LTS operating system.

5.2 Simmulated Annealing

The four important ingredients to use a simulated annealing process, explained in section 2.2, are defined for the following way.

- **System Description** The system configuration is described as a list of edges that will be bent and the complementary list of edges that are cut. Furthermore the facets attached to the edges, as well as the gluetags attached to some of the cut edges.
- **Random Move** To alter the configuration randomly, a random edge of the full eedge list has its weight changed, therefore the minimum spanning tree that is calculated changes.
- **Quantitive Function** To evaluate the system configuration the sum of the overlapping areas is used.
- **Annealing Schedule** The temperature is set to 50.000 as a standard, but can be configured and the cooling rate is set to 1.0, therefore the temperature equals the amount of iterations.

After reading in the mesh into the CGAL Polyhedron_3 data structure preparations are necessary to be able to describe the system. First, the algorithm calculates a dual graph and glue tags, for each edge of the dual graph, as each edge could be a possible cut edge. Furthermore, it sets the temperature, and the cooling rate to 50.000 and 1.0 per iteration, defining the runtime of the annealing process.

The algorithm assigns each edge of the dual graph a random weight and then sorts the list. Then it calculates an initial spanning tree using the edge list. Glue tags are calculated for all edges that will be cut, which are the edges not present in the minimum spanning tree. The algorithm iterates through all possible glue tags and chooses a glue tag if they fulfil the following conditions:

- The edge the gluetag is attached to is a cut edge
- The opposite gluetag was not already added to this edge
- The number of cut edges, that have no gluetag yet is higher than 1 or neither the source face for the gluetag nor the the face the gluetag targets have a gluetag yet

The algorithm discards all glue tags that do not fulfil these conditions, as they are not needed for this unfolding. These conditions ensure that the reconstructed model will be stable when glued together without faces being loose, but they also minimize the number of glue tags to reduce time and effort on reconstruction.

After these steps, the algorithm starts unfolding the triangles and the glue tags alike. It calculates the area of both triangle-triangle overlaps, glue tag-triangle overlaps and glue tag-glue tag overlaps. This approach proposes to weigh triangle-triangle overlaps with a factor of 100, so the algorithm prefers to unfold triangles without overlaps first and afterwards solving glue tag related overlaps. This step is used to optimize the calculation time as well as to influence the evolving graph. The algorithm focuses more on the resolve of triangles, as their overlap area is higher weighted. It is done because a configuration that does not have any glue tag overlaps, but still has triangle overlaps is less optimal than the other way around. If no triangle-triangle overlaps occur in a configuration and only glue tags overlap are left, these have a chance to be resolved for example by changing which glue tags are used or by post processing the size of the gluetags.

The calculated area is then used as the energy of the graph, which we try to minimize in this process. This initial configuration is saved as the best configuration. For each following iteration before these steps are repeated a random edge will be chosen and assigned to a new random weight, which possibly changes the outcoming minimum spanning tree. In the iterative generations and evaluation of configurations also the calculation of the energy differs.

First, the algorithm unfolds the triangles and evaluates the energy of the graph based on the sum of the overlapping area. If and only if the triangles are unfolded without

overlaps, it unfolds the glue tags next. This approach saves computation time, as if the triangles do not unfold without overlaps, no solution is found.

If the energy of the new configuration is smaller or equal to the energy of the previous configuration, the new configuration will be set as the best configuration. If not the configuration is treated probabilistically. If a uniformly distributed random number is smaller than $P(\Delta E) = 1 - e^{-(T+E)/(T_{max})}$ the new configuration will be accepted as the best configuration. This condition should ensure that the algorithm does not get stuck in a local minimum, which would mean that it would not find an optimal global solution.

At the end of every iteration, if a new best configuration was found the new configuration will be visualized using OpenGL, which means that it only changes and impacts calculation time of each iteration if a better configuration was found.

The annealing process ends if it does not find any overlaps or if the temperature reaches the minimum, which means that it ends without finding a solution for this problem. Advantages and Disadvantages using this random walk approach will be discussed in section 6.3.

CHAPTER 6

Results and Evaluation

6.1 Results

The following figures show the resulting unfolding of the implementation. The glue tags are visualized in the 3D-Model and the unfolding, furthermore, the minimum spanning tree (green lines) and cut-edges (red lines) are visualized in the 3D-Model.

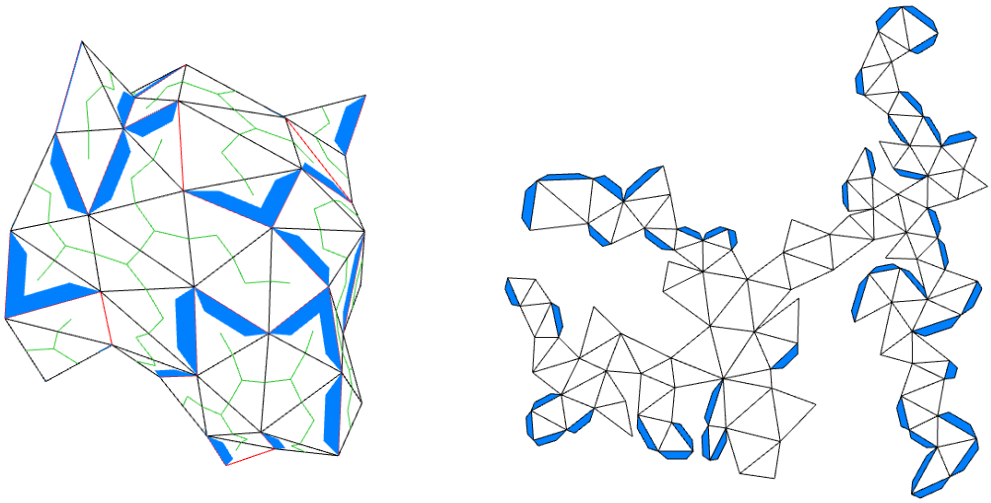


Figure 6.1: 3D Model of a tiger head with 112 faces and the coresponding unfolding.

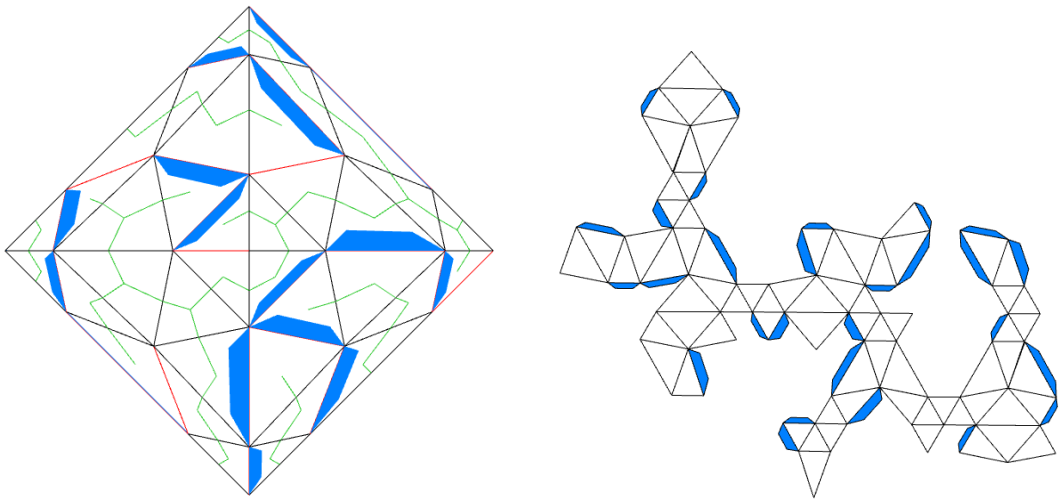


Figure 6.2: 3D Model with 72 faces and the coresponding unfolding.

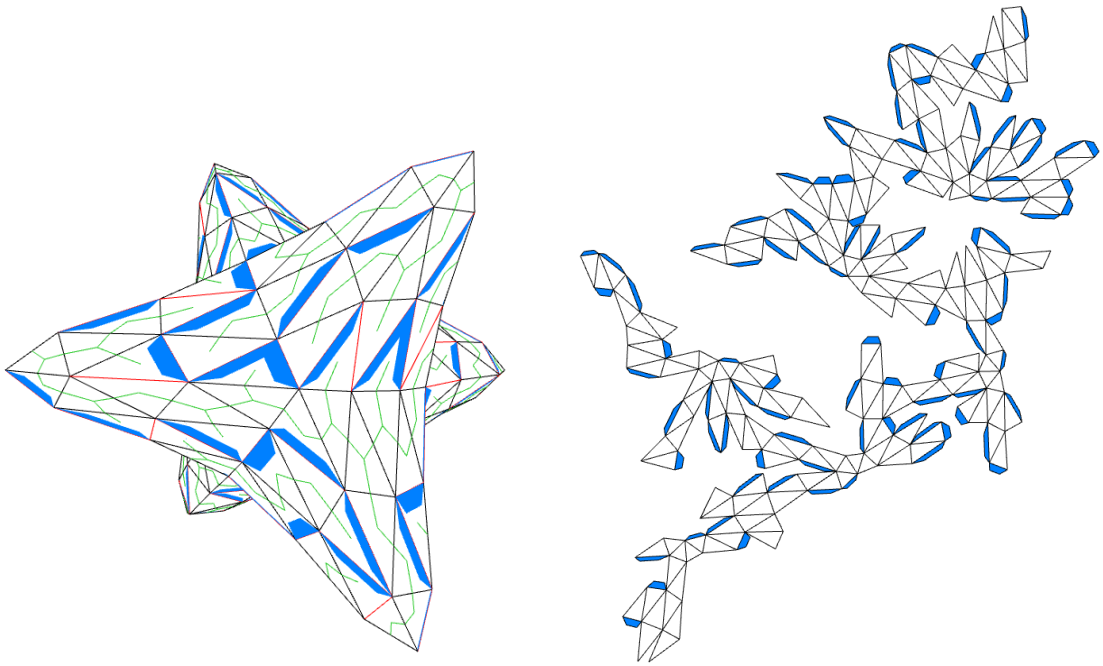


Figure 6.3: 3D Model of a star with 216 faces and the coresponding unfolding.

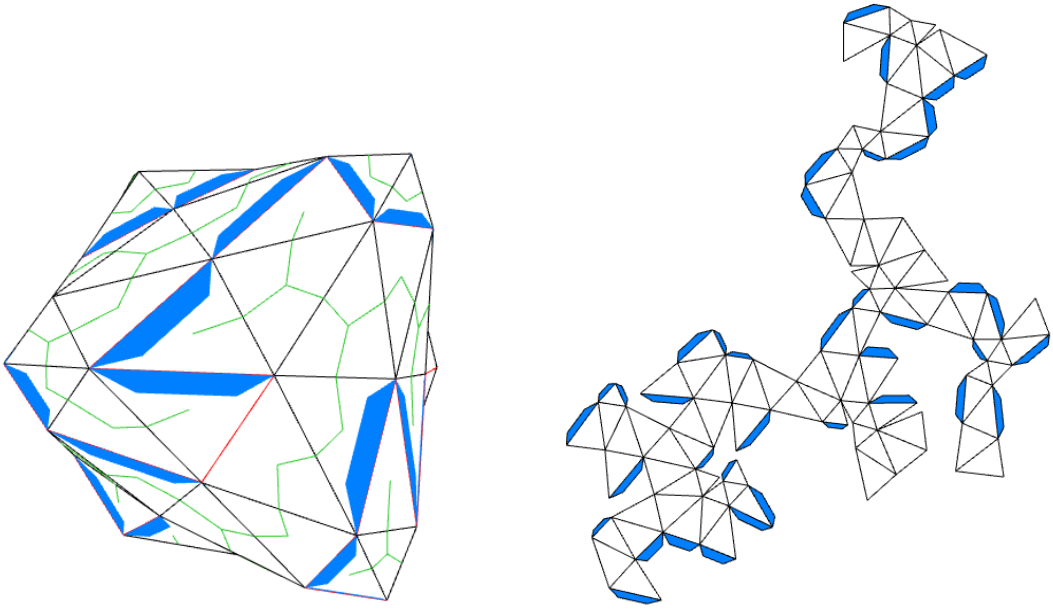


Figure 6.4: 3D Model of a star with 96 faces and the coresponding unfolding.

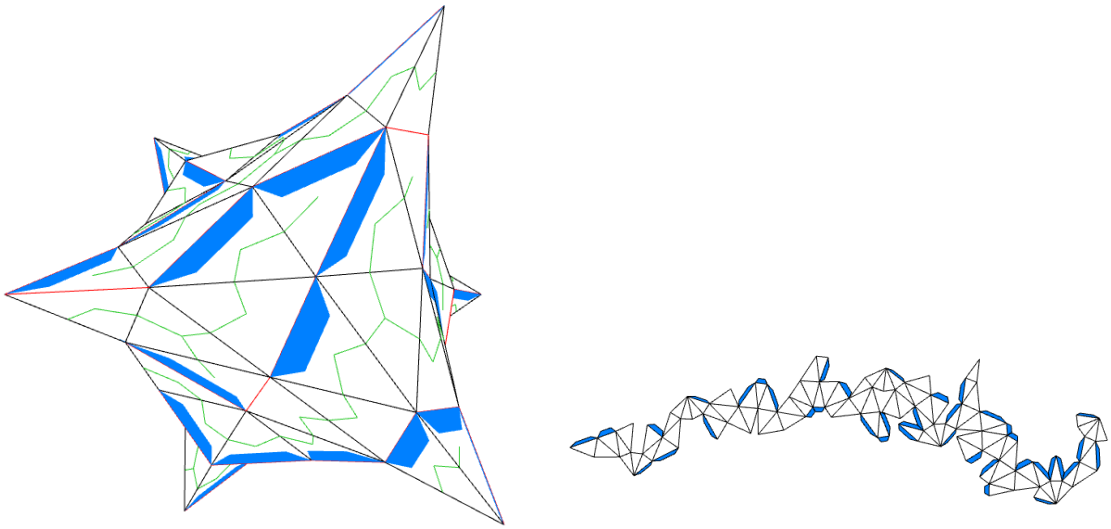


Figure 6.5: 3D Model of a star with 96 faces and the coresponding unfolding.

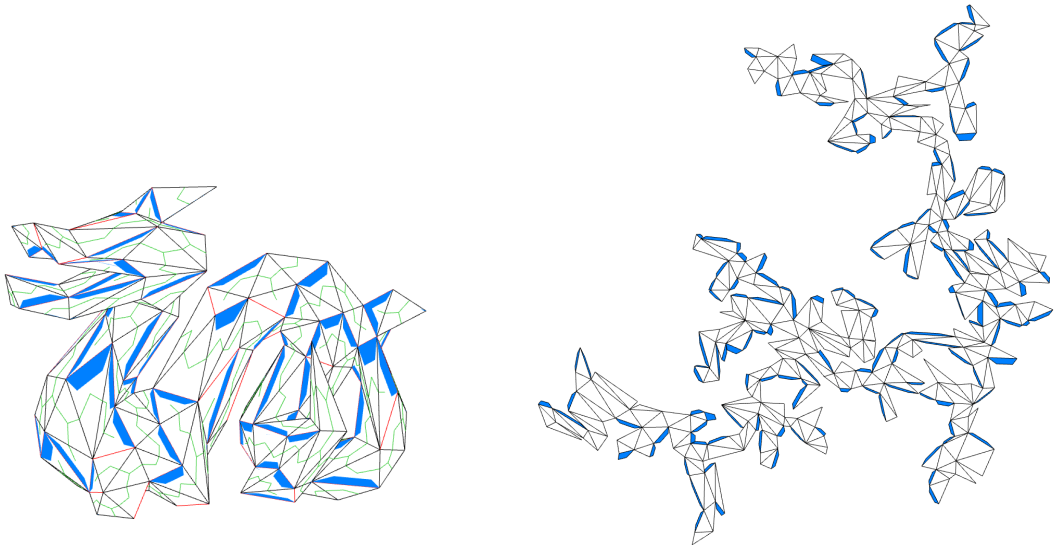


Figure 6.6: 3D Model of a dragon with 344 faces and the coresponding unfolding.

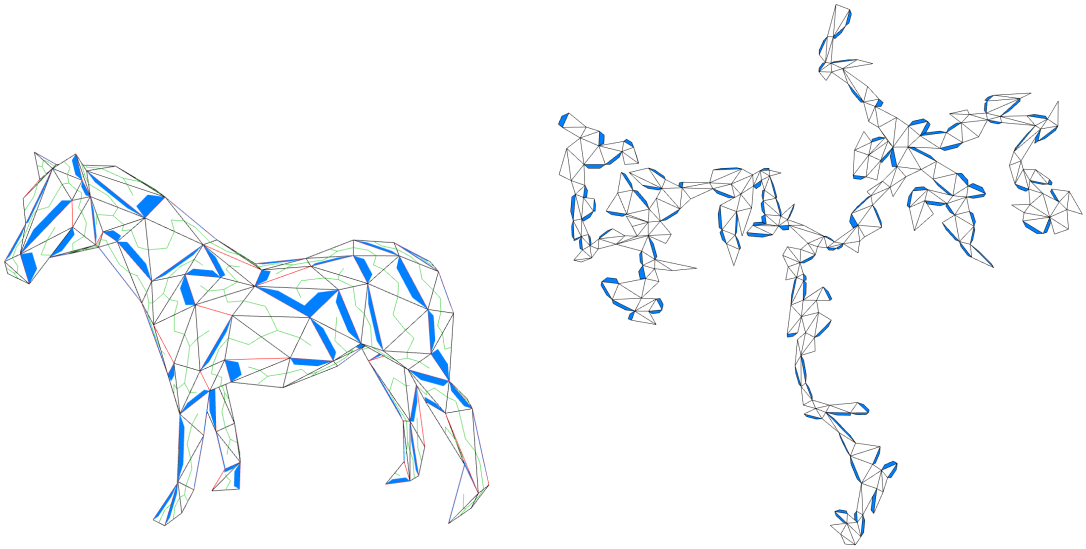


Figure 6.7: 3D Model of a horse with 312 faces and the coresponding unfolding.

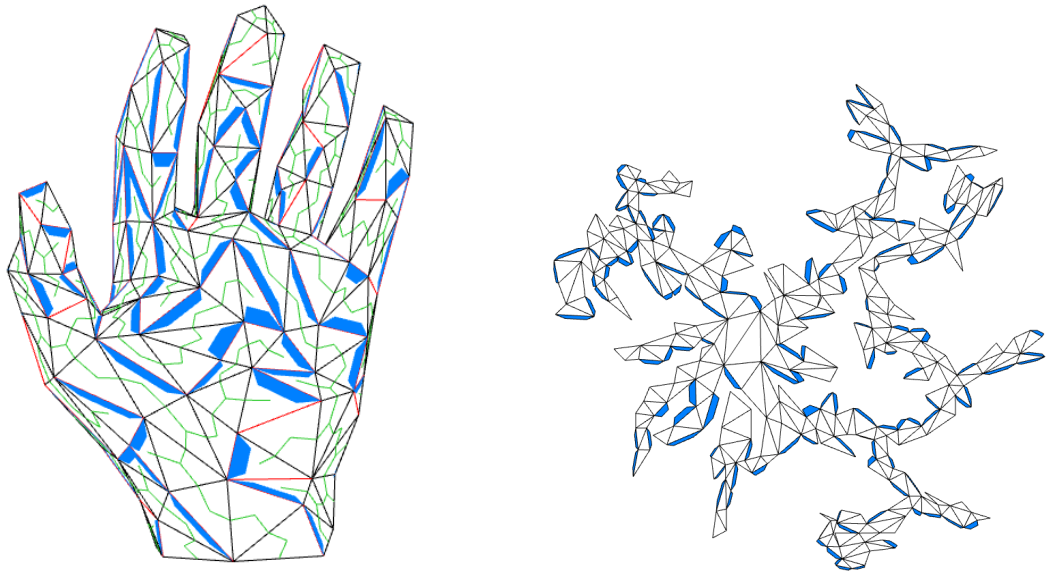


Figure 6.8: 3D Model of a hand with 336 faces and the coresponding unfolding.

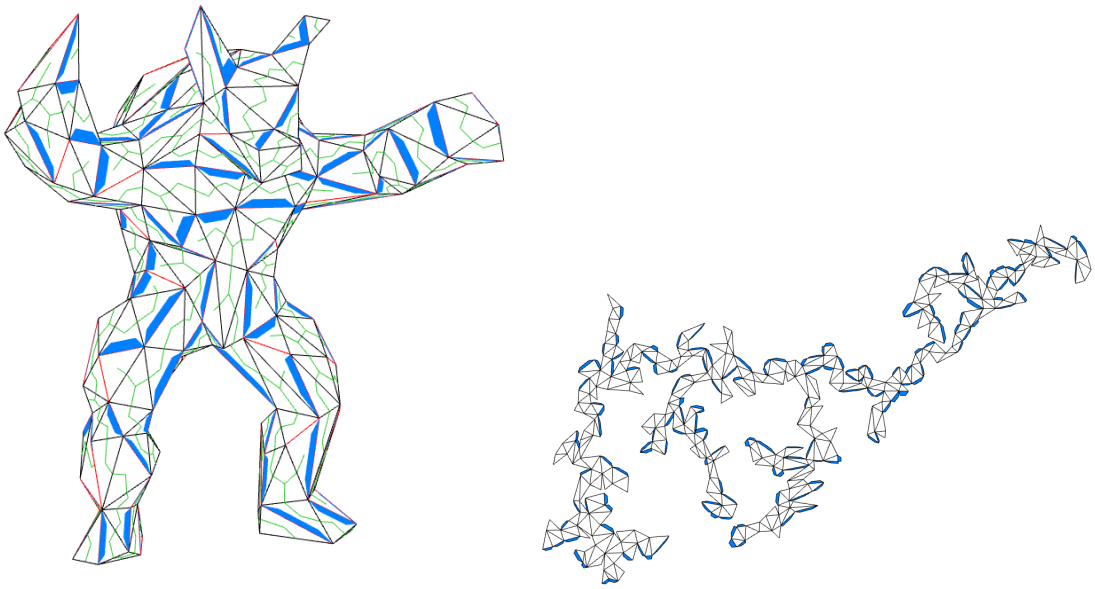


Figure 6.9: 3D Model of an armadillo with 386 faces and the coresponding unfolding.

6.2 Performance

To evaluate the algorithm and its implementation tests were run with a variety of models. The results in table 6.1 were observed using a glue tag that is defined as a trapezoid with the base being the edge of the to-glue triangles and the top being one half of the base with a maximum height of one-fifth of the size of the triangle being targeted by the glue tag. The iterations are capped at a maximum of 100000.

Model		Faces	Time to Unfold (s)
Octa		8	0
Icosa		20	0
Star		24	8
Star-Sqrt3	(Fig. 6.2)	72	31
Star-4Split		96	435
Star-Loop	(Fig. 6.4)	96	137
Star-Butterfly	(Fig. 6.5)	96	1047
Tiger	(Fig. 6.1)	112	65
Kitten		122	48
Moneybox-128		128	160
Bunny-128		128	103
Moneybox-196		196	324
Star-PNsplit	(Fig. 6.3)	216	625
Snail-286		286	1315
Horse	(Fig. 6.7)	312	946
Hand	(Fig. 6.8)	336	1377
Dragon	(Fig. 6.6)	344	1292
Bunny-348		348	976
Luigi		356	686
Armadillo	(Fig. 6.9)	386	730
Pooh		392	957
Moneybox-392		392	2200
Gear		508	-
Cat		702	-
Fish		950	-
Mannequin		1376	-

Table 6.1: Table showing the unfolding performance for different models

Performance is not only influenced by the number of faces, but also by the size of glue tags. The bigger the glue tags are, the more iterations are necessary to find an unfolding, or an unfolding might no longer be possible. Due to the random walk, the time to find an unfolding is not depending on the number of faces. Table 6.1 shows that not only the number of faces influence the time it takes to find a solution, but it also depends on how and which edges are changed. For models marked with a - in the table 6.1 no unfolding was found within 100000 iterations.

6.3 Limitations

Multiple factors limit the suggested approach. The target of the approach is to solve the problem only considering the global optimum, disregarding local overlaps. Therefore the latter are hard to resolve, as they are not explicitly targeted.

Another limitation is that the glue tags that are necessary cannot be calculated beforehand. Neither the amount nor the position of the glue tags can be calculated, without using a heuristic approach or brute force, whereas brute force is not feasible as the performance even with small models is abysmal.

Another limitation that does not have only negative consequences is the random-walk approach used in the simulated annealing. A random edge is chosen and changed in each iteration, which might not change the minimum spanning tree that is generated. It can happen that an edge that is causing an overlap does not change for many iterations.

6.4 Parameterization

The algorithm is optimizable by changing multiple parameters without changing the approach itself.

Gluetag size is the most influential parameter that can be changed. Depending on the size of the glue tag, the solution space is widened or made smaller if the glue tags area increases.

Probabilistic treatment of worse iterations can also be changed to either favour taking a worse iteration more or less often. If the probability is too high to take a worse iteration, the algorithm will not find a solution and might not even get close to a solution if the probability is too low, the chance of getting stuck in local minimums increases.

Number of iteration can be controlled by changing the cooling down rate as well as the maximum and minimum temperature, therefore changing the time the algorithm has to find a solution.

By tuning these parameters, the results can be influenced to get better performance or make finding an unfolding possible.

CHAPTER 7

Conclusion

7.1 Summary

Calculating an unfolding using a minimum spanning tree approach is possible and can yield excellent performance for smaller meshes. As meshes have increased numbers of faces and glue tags increase in size, the worse the algorithm is performing. The reconstruction is almost impossible without any visual cues on which edges should be folded or glued first, but disregarding this factor as it can be added with this approach too, a more significant setback is that the unfolding is more or less random and structures of the 3D-Model might not be well conserved into the unfolding. All in all, the approach is simple to implement as no sophisticated algorithms are needed, and it yields results in an acceptable amount of time for models with less than 200 faces.

7.2 Future Work

The quality of the unfolding could be improved, by not only measuring the unfolding by the overlap area but also considering other factors, like keeping structures of the mesh together to help users with reconstructing the models.

Performance can be improved, if glue tags are post-processed to change their shape if the overlap area is rather small. With this improvement, the computation time can be reduced significantly, without impacting other parts of the approach.

Once an unfolding with glue tags for a model is found, a greedy algorithm could be used to minimize the number of glue tags necessary. Currently, the amount of glue tags is determined by the order the glue tags are added. Therefore it can result in using more glue tags than necessary for the particular unfolding.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AMOT90] Ravindra K Ahuja, Kurt Mehlhorn, James Orlin, and Robert E Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM (JACM)*, 37(2):213–223, 1990.

[BM95] Stephen P Brooks and Byron JT Morgan. Optimization using simulated annealing. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 44(2):241–257, 1995.

[CT76] David Cheriton and Robert Endre Tarjan. Finding minimum spanning trees. *SIAM Journal on Computing*, 5(4):724, 1976.

[DA91] Anton Dekkers and Emile Aarts. Global optimization and simulated annealing. *Mathematical programming*, 50(1-3):367–393, 1991.

[DDF14] Mirela Damian, Erik D Demaine, and Robin Flatland. Unfolding orthogonal polyhedra with quadratic refinement: the delta-unfolding algorithm. *Graphs and Combinatorics*, 30(1):125–140, 2014.

[DFO07] Mirela Damian, Robin Flatland, and Joseph O'rourke. Epsilon-unfolding orthogonal polyhedra. *Graphs and Combinatorics*, 23(1):179–194, 2007.

[GFR94] William L Goffe, Gary D Ferrier, and John Rogers. Global optimization of statistical functions with simulated annealing. *Journal of econometrics*, 60(1-2):65–99, 1994.

[GY04] Jonathan L Gross and Jay Yellen. *Handbook of graph theory*. CRC press, 2004.

[KGJV83] S Kirkpatrick, CD Gelatt Jr, and MP Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598), 1983.

[Kru56] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[MGPO89] Miroslaw Malek, Mohan Guruswamy, Mihir Pandya, and Howard Owens. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research*, 21(1):59–84, 1989.

[MS04] Jun Mitani and Hiromasa Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. In *ACM transactions on graphics (TOG)*, volume 23, pages 259–263. ACM, 2004.

[PBARCC90] J Pannetier, J Bassas-Alsina, J Rodriguez-Carvajal, and V Caignaert. Prediction of crystal structures from crystal chemistry rules by simulated annealing. *Nature*, 346(6282):343, 1990.

[SDJ95] Jon M Sutter, Steve L Dixon, and Peter C Jurs. Automated descriptor selection for quantitative structure-activity relationships using generalized simulated annealing. *Journal of chemical information and computer sciences*, 35(1):77–84, 1995.

[SH74] Ivan E Sutherland and Gary W Hodgman. Reentrant polygon clipping. *Communications of the ACM*, 17(1):32–42, 1974.

[She75] Geoffrey C Shephard. Convex polytopes with convex nets. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 78, pages 389–403. Cambridge University Press, 1975.

[SP11] R Straub and H Prautzsch. Creating optimized cut-out sheets for paper models from meshes. 2011.

[ŠVV17] Martin Šlapák, Josef Vojtěch, and Radek Velc. Automated numerical calculation of sagnac correction for photonic paths. *Optics Communications*, 389:230–233, 2017.

[The19] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.14 edition, 2019.

[TWS+11] Shigeo Takahashi, Hsiang-Yun Wu, Seow Hui Saw, Chun-Cheng Lin, and Hsu-Chun Yen. Optimized topological surgery for unfolding 3d meshes. In *Computer graphics forum*, volume 30, pages 2077–2086. Wiley Online Library, 2011.

[XKKL16] Zhonghua Xi, Yun-hyeong Kim, Young J Kim, and Jyh-Ming Lien. Learning to segment and unfold polyhedral mesh from failures. *Computers & Graphics*, 58:139–149, 2016.