Optimising 3D Mesh Unfoldings with Additional Gluetags using Simulated Annealing

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: Ph.D. Hsiang-Yun Wu

Wien, 1. März 2019

Thorsten Korpitsch	Hsiang-Yun Wu

Optimising 3D Mesh Unfoldings with Additional Gluetags using Simulated Annealing

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: Ph.D. 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-Mesh Unfolding 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, Ph.D. 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 eines 3D Netze in ein 2D Netze. Diese Technik kann verwendet werden, um Papercraft-Modelle zu erstellen, dabei werden 3D-Objekte mit einem Papier oder papierähnlichem Material rekonstruiert. Da die Rekonstruktion von Modellen schwer sein kann, benötigen Anwender Indikatoren, welche Flächen miteinander verklebt werden sollen. In dieser Arbeit werden sogenannte Gluetags vorgestellt, die Anwendern Platz geben, um Klebstoff aufzutragen, um die Rekonstruktion zu erleichtern. Das Hinzufügen dieser Gluetags erhöht die Schwierigkeit überlappungsfreie Entfaltungen zu finden, die aus einem einzigen Stück Papier ausgeschnitten werden können. Dabei erhöht sich die Anzahl der möglichen Entfaltungen, während der Lösungsraum schrumpft. Ein Minimum Spanning Tree Ansatz wird verwendet, um mögliche Entfaltungen zu berechnen, während simuliertes Glühen verwendet wird, um die Entfaltung zu optimieren und eine Lösung ohne Überlappungen zu finden. Quantitative Experimente deuten darauf hin, dass der vorgeschlagene Ansatz schnelle Ergebnisse für kleinere Netze. Für größere Netze werden Ergebnisse innerhalb eines größeren Zeitrahmens geliefert, bei diesen zeigen aber auch eindeutige zeitliche Beschränkungen auf.

Abstract

3D Mesh Unfolding is the process of transforming a 3D mesh into a 2D planar patch. This technique can be used to create papercraft models, where 3D objects get reconstructed from planar paper or paper-like material. As the reconstruction of unfolded models can be very hard, users need indicators of which faces have to be glued together. In this thesis, Gluetags are introduced to give users extra space to apply glue to ease the reconstruction. The addition of these Gluetags increases the difficulty of finding overlap-free unfoldings that can be cut out of a single piece of paper to reconstruct the model. The amount of possible unfoldings increases while the solution space shrinks when Gluetags are added. A minimum spanning tree approach is used to compute possible unfoldings, whereas simulated annealing is used to find an unfolding with no overlaps. Quantitative experiments suggest that the proposed method can yield fast results for smaller meshes. Results for larger meshes are achievable within an increased timeframe, but they also show time limitations for this approach.

Contents

CHAPTER 1

Introduction

This chapter gives a brief overview of what papercraft is, and insights into the background of this work. Furthermore, it explains the motivation and goals of this thesis. Lastly, it discusses shortly the results and describes how the remainder of this thesis structured.

1.1 Background

Papercraft is a widely popular art of creating two or three-dimensional objects from cardboard or paper, as seen in figure 1.1. The models that are created range from simple ones, like paper aeroplanes, to elaborate models of buildings or districts for city planning.

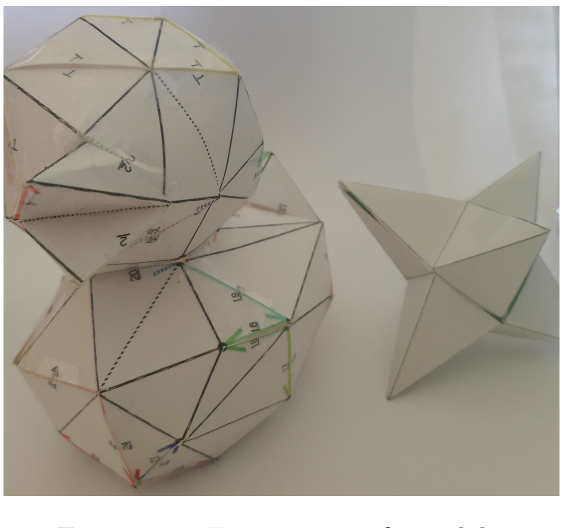


Figure 1.1: Two papercraft models.

.

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 single 2D patch or multiple 2D patches. Takahashi et al. [TWS$^+$11] found that unfolding into a single 2D patch avoids the problem of seeking the correspondences between the boundary edges of different patches when merging them. The resulting mesh can, for example, be printed onto paper and then reconstructed into the 3D model. Due to two main problems occurring during the process of unfolding this task is highly complex. 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

The reconstruction of a model can be very hard, even with indicators that show which faces should be glued together, like the solution presented by Takahashi et al. [TWS$^+$11]. Glueing the faces together if they are tiny is still a hard task, because there is almost no space to apply glue to. This thesis introduces *Gluetags* to make the reconstruction of models easier. These Gluetags add small faces on edges that are cut and give users space to apply glue. Adding Gluetags, 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. The inclusion of Gluetags makes the problem more difficult as the solution space for an unfolding without overlaps shrinks, whereas the search space of possible unfoldings and possible Gluetag positions increases.

1.3 Goal

The goal of this thesis is to explore the addition of Gluetags to cut edges. This thesis proposes the addition of Gluetags in advance to the 3D-Model before the unfolding process starts. During the unfolding Gluetags are treated as part of the original mesh. Hence the same unfolding algorithms can be used without further changes. Steps of the process are visualised and allow users to interact with both the 3D-Model and the final planar patch.

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 in size. Additionally 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 within an increased timeframe. Meshes over 700 faces cannot be consistently resolved. Another factor impacting the time to solution and quality of the result is the size of Gluetags. An increased Gluetag size makes finding overlap free unfolding less likely. The size of Gluetags can be adjusted to counteract the increase in time needed for unfolding meshes with a higher face count.

1.5 Structure

The remainder of this thesis is structured as follows. In chapter 2, we provide an overview of previous findings related to 3D Mesh Unfolding and highlights differences to the proposed approach in this thesis. It also discusses the theoretical background and related work for simulated annealing, that optimises the search for unfoldings. Chapter 3 describes the concepts of dual graphs and minimum spanning trees, and further describes the data structure for the suggested approach. The next chapter, 4, gives an overview of the processing pipeline, that computes an unfolding and describes necessary steps in detail. 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 evaluates its performance and limitations. Finally, chapter 7 summarises the findings and provides an outlook on future work.

CHAPTER 2

Related Work

This chapter focuses on previous work done on the topic of optimising the unfolding of 3D Meshes but also points out the key differences to the approach proposed in this thesis. Furthermore, it presents the optimisation technique of simulated annealing.

2.1 Optimised Unfolding of 3D Meshes

Mesh Unfolding has different applications, like creating papercraft models [TWS$^+$11, SP11] and the creation of models from self-folding materials [FTS$^+$13, Tib14]. Many different unfolding techniques have been explored. Theoretical approaches for unfolding meshes have been intensively explored [She75], while other authors focus on different kind of meshes, for example, orthogonal polyhedra [XKKL16, DFO07, DDF14].

Takahashi et al. [TWS$^+$11] use a genetic-based algorithm to find unfoldings. They unfold 3D Meshes based on polyhedron models into a single patch, which is still a well-known open problem. Their heuristic approach tries 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.

This thesis, on the other hand, uses the meta-heuristic simulated annealing approach to find unfoldings. Furthermore, the key concept of the proposed approach is to find unfoldings based on minimum spanning trees that can be calculated from the dual graph of the mesh.

Straub et al. [SP11] explored the unfolding and adding Gluetags to an unfolded mesh. 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 optimise the cutout and to resolve overlaps. Gluetags that have been added to an unfolding are optimised, i.e. changed in size to fit and non-overlap, after an unfolding is found.

In this thesis, the proposed algorithm computes all possible Gluetags beforehand and unfolds them together with the original mesh. Only necessary Gluetags are considered for each unfolding. Furthermore, the Gluetags are not treated special, but as part of the mesh and they are therefore not post-processed and changed.

Mitani and Suzuki [MS04] propose a different approach. In their paper, they describe the production of unfoldings by using strip-based approximation. They propose to segment meshes into parts of easily reconstructible segments. This is achieved using 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 focuses on large mesh models where they merge regions between 60 and 250 triangles.

In this thesis, we suggest an authentic reconstruction of the original model. Therefore no mesh simplification methods are used and the reconstructed model is authentical to the original model.

In contrary to Chang et al. [CY17], where the introduction of additional cuts into the polyhedra is proposed, the suggested approach in this thesis only cuts along edges that are later glued together again, therefore the original model is not changed during unfolding.

2.2 Optimisation Techniques

As the unfolding of 3D Meshes is considered an NP-complete problem[HE12] optimisation techniques are necessary to minimise the time needed to find a solution. Many optimisation techniques are well explored, like greedy algorithms[DT96] or heuristic optimisation techniques[LES08]. This thesis proposes to use simulated annealing as the optimisation technique for the problem of finding optimal unfoldings. Compared to hill climbers, simulated annealing is less likely to get stuck in local optimums, which often appear in problems akin to 3D Mesh Unfolding. Furthermore, simulated annealing is easy to implement and very configurable, as later explained in this section.

Simulated annealing is a well-known optimisation 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 minimum value of a function, which is called the cost function, which has many independent variables. A typical example that simulated annealing is used is the travelling salesman problem [MGPO89], where the most effective route between different cities has 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 value E of configuration P is assumed to be minimal.

The process has to be extended by another step to counteract 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. If $R < P(\Delta E)$ then the new configuration is accepted, where R is a uniformly distributed random number within the range of $[0, 1]$. Otherwise, it discards the new configuration. As T decreases over time, it gets less likely to accept worse configurations and the algorithm execution comes to an end.

To summarise, simulated annealing consists of four essential components. 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 quantitative 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 time. This includes a cooling-rate used to stop the process, which tries to emulate the cooling down of metal which it can only be formed while it is hot.

CHAPTER 3

Definition of Data and Key Concepts

This chapter describes the properties of the data that is processed in the proposed implementation. It also provides definitions for the key concepts used in this thesis.

3.1 Data Representation of the 3D Mesh Model

The original mesh data has to meet the following requirements to be applicable for the suggested approach.

- Mesh data is in the Object File Format (.off)
- Mesh data is triangulated
- No duplicated vertices, edges or zero-area faces
- Mesh has no holes
- All faces are connected

The data is read from a file and saved into the CGAL [The19] data structure Polyhedron_3. 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 linked together, therefore enabling iterating through all halfedges of a face. With this data structure, all information is available enabling the application of the approach suggested in this thesis.

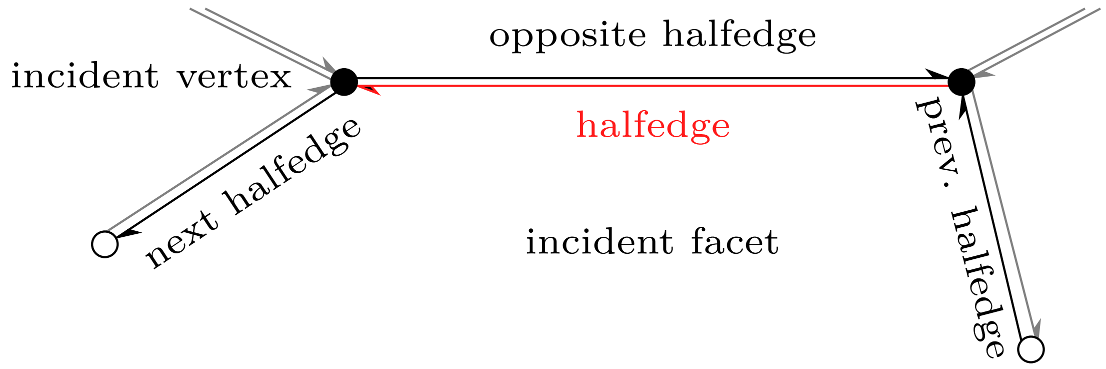


Figure 3.1: CGAL Polyhedron_3 datastructure visualised. Adapted from the CGAL user manual [The19].

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 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, which means the dual graph contains all edges, whether they are cut or bent, of the mesh model. A graph has only one dual graph, and the calculation of it can be done very efficiently, which makes it a very compelling data structure to use for 3D mesh unfolding.

3.3 Cut- and Bend Edges

In this thesis, each edge of the dual graph can represent a cut edge or a bend edge. A cut edge is an edge that is cut to be able to unfold the mesh and can be glued back together later. Contrary to that, a bend edge is not cut but bent in the process of reconstruction.

3.4 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]. An option to compute a minimum spanning tree is to first sort the edges by their weight and then one edge after another is added to the minimum spanning tree. If an edge makes the graph cyclic, this edge is removed again. After this process, the minimum spanning tree is defined by the edges that are

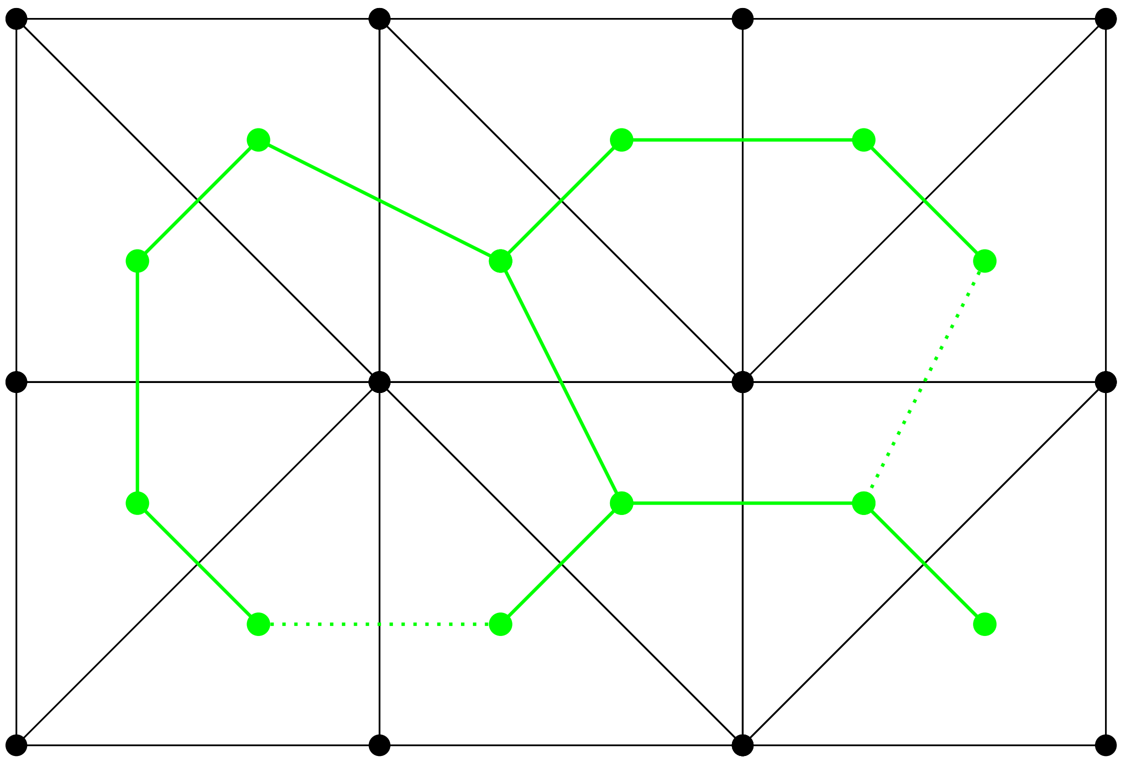


Figure 3.2: (black) Shows a graph with undirected edges. (green solid and dotted) Shows the graphs complete dual graph. (green solid) Shows a minimum spanning tree of the dual graph.

not discarded. Many simple algorithms for computing minimum spanning trees are available [Kru56, AMOT90]. A minimum spanning tree is therefore acyclic by definition and can be used to find an unfolding, as an unfolding must not contain cycles as well.

If the minimum spanning tree is calculated from the dual graph, the edges that are part of the minimum spanning tree are bend edges. All other edges, which are not in the minimum spanning tree, are cut edges. Therefore, due to its simple calculation, a minimum spanning tree in combination with the dual graph is a good tool to calculate possible unfoldings.

A weighted graph, where no edges have the same weight, has exactly one minimum spanning tree. This is a problem since the dual graph does not inherently have weights assigned to edges. Weights need to be assigned to each edge of the dual graph. However, the weights cannot be constant values, as it is done for the travelling salesman problem, because not all minimum spanning trees will yield an unfolding without any overlaps. As initial weights, this thesis proposed to random values between $(0, 1)$.

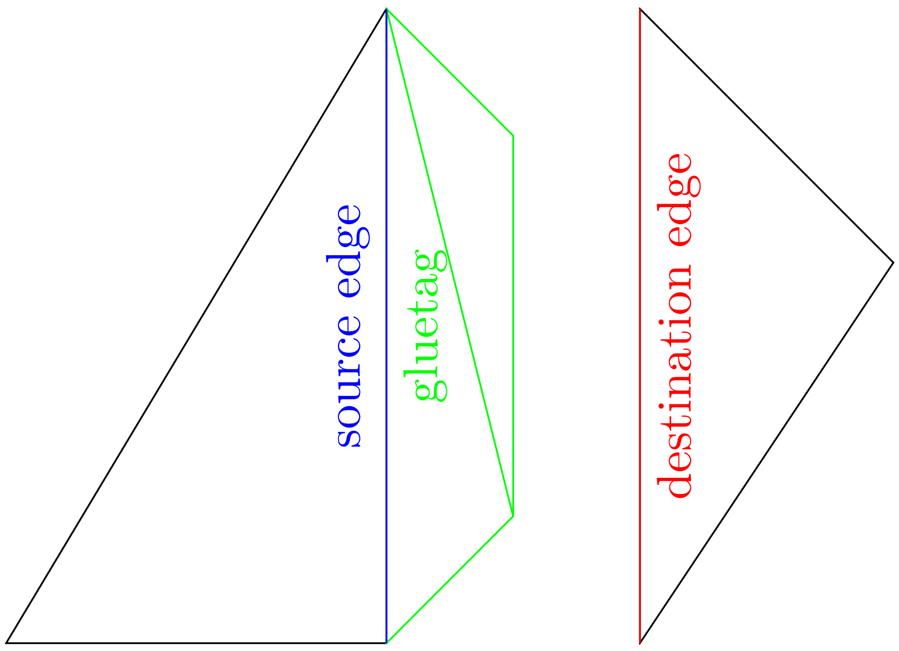


Figure 3.3: Exemplary trapezoid Gluetag (green) attached to its source triangle edge (blue) and the triangle it will be glued to (red).

3.5 Gluetag

A Gluetag is a space that users can use to apply glue. It can have different forms, for example, the shape of a trapezoid, as shown in figure 3.3. As seen in figure 3.3, for each Gluetag exists one source and one destination edge. As a result there exist two valid positions for each that can be swapped to point into the other direction, therefore switching the source and target edge. In this thesis, we propose trapezoid-shaped Gluetags which are put on one side of a cut edge at most. Another advantage of the trapezoid shape is that the Gluetags are treated the same way as the triangles of the original mesh, since each trapezoid consists of two triangles.

3.6 Unfolding

An unfolding is defined as the 2D representation of the 3D Model, after unfolding. It is created by unfolding faces after each other, according to the minimum spanning tree. The exact order of which face is unfolded first can be neglected as the minimum spanning tree defines exactly one unfolding. The quality of an unfolding is defined by the overlapping area in this thesis, since distortion cannot appear due to the used algorithm. In this thesis, an optimal or correct unfolding is therefore defined as an unfolding without distortion of faces and no overlapping areas. The approach suggested in this thesis avoids distorting faces to make an accurate reconstruction possible. Additionally it is necessary to guarantee that areas do not overlap to enable the creation of a cut-out using a single piece of paper.

CHAPTER 4

Methodology

Figure 4.1 gives an overview of the solution proposed in this thesis. It shows each step from loading a 3D Model to visualising the resulting unfolded model. 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 Gluetags are calculated, each targeting one facet of the edge and having its source on the other facet, see Figure 3.3. 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 sub-chapters, each step is described in more detail, whereas the simulated annealing process is the focus of chapter 5.

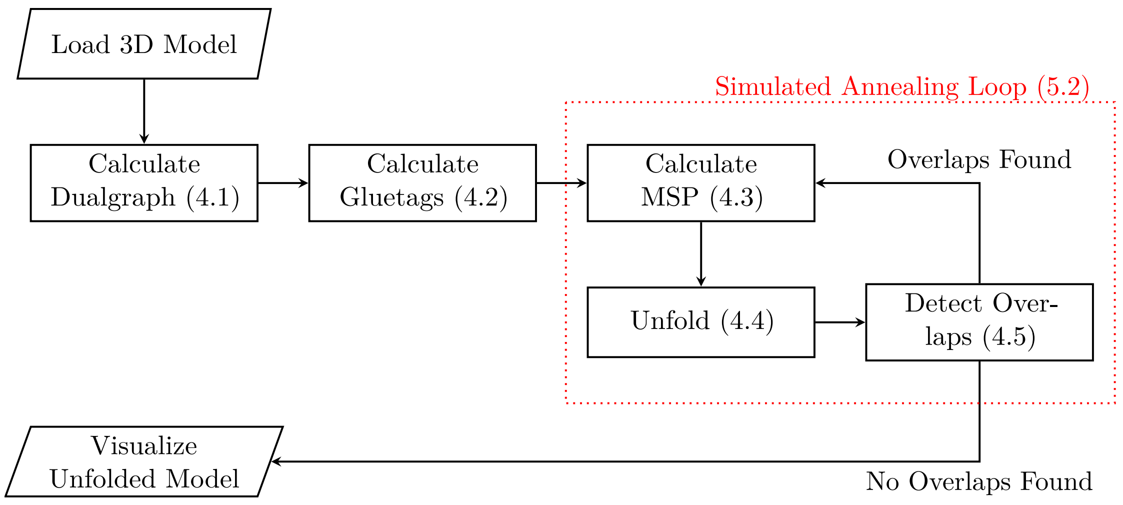


Figure 4.1: Overview of the 3D Mesh-Unfolding Process.

4.1 Calculating the Dualgraph

As the dual graph, described in chapter 3, is calculated from the original mesh, which does not change during the unfolding process, the dual graph is calculated only once in the beginning, 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 initialised with a weight. In this thesis, the weight is initialised and changed throughout the algorithm using a random number to be able to utilise the random walk approach of simulated annealing.

4.2 Calculating the Gluetags

The second step in the pipeline shown in Figure 4.1 is to calculate Gluetags. For each edge of the dual graph, a Gluetag is calculated for both facets connected by this edge. The endpoints of an edge are the base of a Gluetag. As Gluetags, an example shown in Figure 3.3, can vary in shape and size, this thesis proposes Gluetags in the shape of a trapezoid as it brings a few advantages.

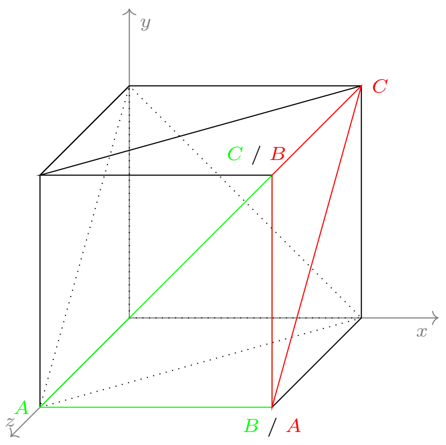
Two triangles define a trapezoid. Therefore algorithms that are applied to triangles of the mesh can be applied to the Gluetags without altering them. As the top of the trapezoid is smaller than the base, Gluetags that are placed next to each other are less likely to overlap compared to rectangular Gluetags. The algorithm calculates the height of the Gluetag depending on the targeted facet so that it takes up a maximum of 20 percent of the targeted facets space. The height of the Gluetag is an experimental value, as well as the shape, to provide users enough extra space, while not shrinking the solution space further than necessary.

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

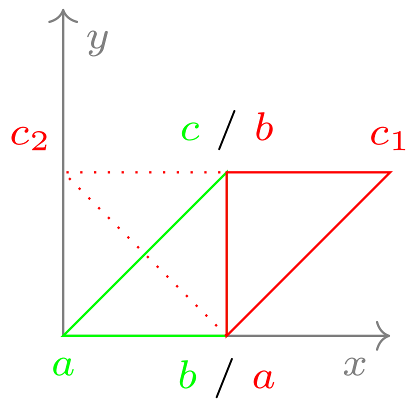
4.3 Calculating a Minimum Spanning Tree

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 edge last added is removed and inserted into a list containing the cut edges. Otherwise,



(a) 3D Mesh of a box.



(b) Unfolding of the first two triangles.

Figure 4.2: 3D model of a box and the unfolding of the first two triangles.

the edge 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.

4.4 Unfolding the 3D Mesh using the Minimum Spanning Tree of its Dual Graph

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, which is on index zero of the adjacency list, the green triangle (A, B, C) in figure 4.2, that is unfolded 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 as $\overline{AB} = \overline{ab} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}$.

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. Two solutions are possible for c, as it can be placed on either side of the edge $(a - b)$.

$$\begin{aligned} s &= \frac{\|(B - A) \times (C - A)\|}{(\overline{AB})^2} \\ d &= \frac{(B - A) \cdot (C - A)}{(\overline{AB})^2} \\ c_x &= a_x + d(b_x - a_x) - s(b_y - a_y) \\ c_y &= a_y + d(b_y - a_y) + s(b_x - a_x) \end{aligned} \tag{4.1}$$

The first candidate solution for $c_1 = (c_x, c_y)$ is defined by the formulas 4.1. The second candidate solution for c_2 can be calculated by the formulas 4.2.

$$\begin{aligned} c_x &= a_x + d(b_x - a_x) + s(b_y - a_y) \\ c_y &= a_y + d(b_y - a_y) - s(b_x - a_x) \end{aligned} \tag{4.2}$$

For the first triangle, either c_1 or c_2 can be chosen, as it only changes in which direction the other vertices are unfolded. 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 $c_{prev} = (a_{x_{prev}}, a_{y_{prev}})$ be the vertex of the previous triangle that it does not share with the current triangle, as seen in figure 4.2. To calculate the side on which the vertex c has to be put can be determined by the formula 4.3.

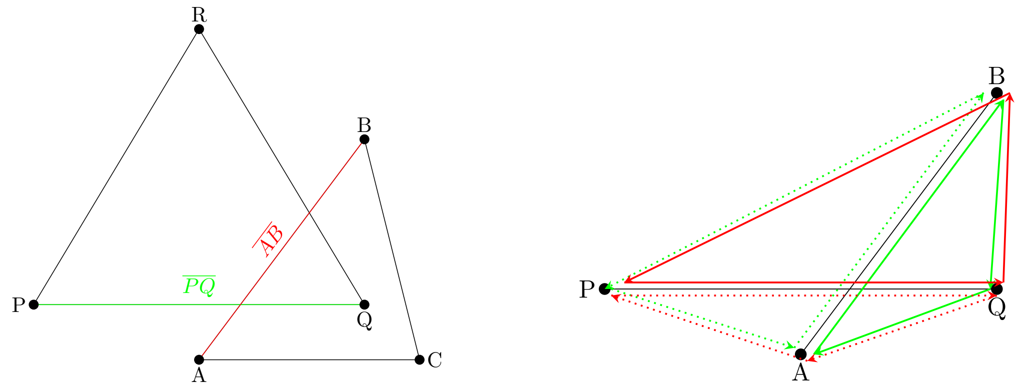
$$f = (c_x - a_x) * (b_y - a_y) - (c_y - a_y) * (b_x - a_x) \tag{4.3}$$

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 $c = c_2$. In case of figure 4.2 the previous vertices that is not shared is left of the shared edge, therefore the vertices c is set on the right side.

The unfolding of Gluetags works in the same way, as the Gluetag consists of two triangles.

4.5 Detecting Overlaps

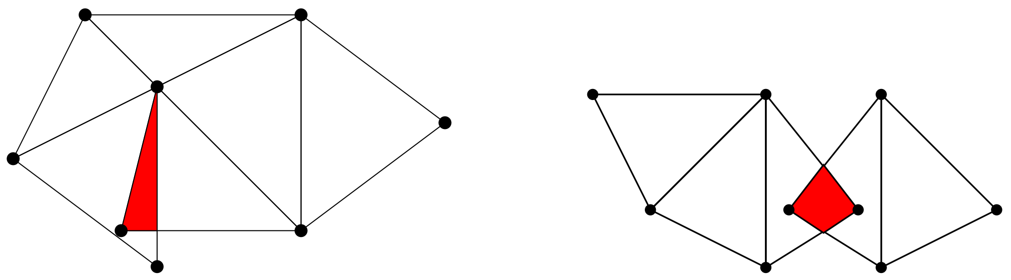
The last step of the simulated annealing loop, as shown in figure 4.1, is to determine the quality of the unfolding. As a measurement for the quality, merely the sum of the



(a) Two triangles $T = (A, B, C)$ and $S = (P, Q, R)$ overlapping each other.

(b) $\overline{AB}$ and $\overline{PQ}$ intersecting each other.

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



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

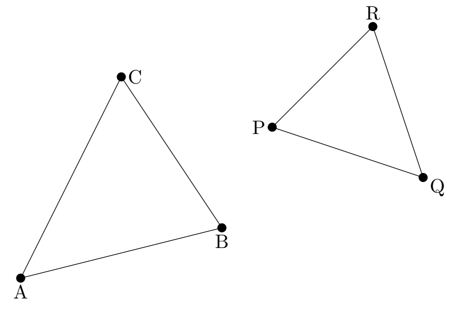
(b) Two triangles overlap each other, resulting in a polygon describing the red overlap area.

Figure 4.4: Overlaps of triangles resulting in different overlapping areas.

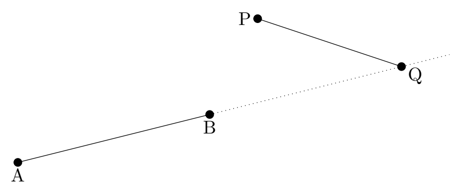
overlapping areas is used. Overlaps, as seen in Figure 4.4, can happen between triangles, triangles and Gluetags or between Gluetags.

The detection is a two-step process. First, only the triangles are checked if they overlap each other. In the second step, if no triangles overlap each other, the Gluetags 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.

Let $T = (A, B, C)$ and $S = (P, Q, R)$ be two triangles in figure 4.3 that might be overlapping each other, defined by their points A, B, C, P, Q, R. Therefore we check the lines $\overline{AB}$ and $\overline{PQ}$, as seen in figure 4.3b, if they intersect with each other. For two line segments, we define a general case for overlaps and a special case. In general the lines intersect only if the points (A, B, P) (dotted green line) and (A, B, Q) (solid green line) have different orientations as well as (P, Q, A) (dotted red line) and (P, Q, B) (solid red line) have different orientations. In figure 4.3, it can be seen that the green lines have

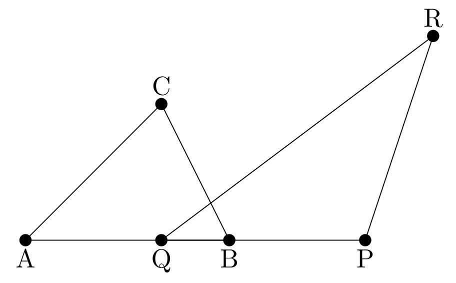


(a) Two triangles $T = (A, B, C)$ and $S = (P, Q, R)$ not overlapping each other.

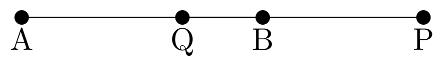


(b) $\overline{AB}$ and $\overline{PQ}$ not intersecting each other.

Figure 4.5: Overlaps of triangles resulting in different overlapping areas.



(a) Two triangles $T = (A, B, C)$ and $S = (P, Q, R)$ overlapping each other.



(b) $\overline{AB}$ and $\overline{PQ}$ are colinear and intersect each other.

Figure 4.6: Two triangles overlapping each other, due to colinear lines.

different orientations and red lines too. Therefore the lines intersect, which means that the triangles overlap each other. This test is repeated for every pair of lines.

Figure 4.5 shows two triangles that do not intersect each other, as (A, B, Q) and (A, B, P) have different orientations, but (P, Q, A) and (P, Q, B) have the same orientation, therefore the triangles do not intersect.

For the special case, the lines intersect, if all four point-triplets are co-linear and the x-projections of (A, B) and (P, Q) intersect and the y-projections of (A, B) and (P, Q) intersect as well, which can be seen in figure 4.6.

The proposed solution calculates for every pair of lines of the two triangles if they intersect. If at least one pair of lines intersect with 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 4.4.

After detecting that an overlap occurs, the algorithm calculates the area of the polygon,

Algorithm 4.1: Sutherland-Hodgman pseudo algorithm. Adapted from [Wik19].

```
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;
```

which describes the overlap, to determine quality of the current configuration, the overlapping area is calculated using the Sutherland-Hodgman Clipping algorithm [SH74]. This algorithm finds the vertices of the intersection polygon created by the two triangles. The overlap can be described as a polygon, and it can be either be defined at least as a triangle, see Figure 4.4a, or a polygon with a degree, see Figure 4.4b, depending on how the triangles overlap each other. Algorithm 4.1 shows the pseudo-code 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] $Area = \frac{1}{2}\left|\sum_{i=0}^{P-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right|$ 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.

CHAPTER 5

Implementation

This chapter specifies on which computer system the proposed approach is developed as well as the libraries that are used. Further, the implementation details of the simulated annealing process are explained.

5.1 Specifications

The system is implemented on a laptop with an Intel Core i7 CPU (4 cores @ 3.3 GHz, 4MB Cache) and 8GB RAM. The source code is written in C++17, and the OpenGL 4.5 library is used for the visualisation of the algorithms step as well as the results. The CGAL 4.13 library is used to read in off-Files and to provide the underlying data structure. The Graphical User Interface (GUI) is developed using the Qt Library 5.13. CMake 3.14 is used to manage the build process, and it compiles on an Ubuntu 18.04.02 LTS operating system.

5.2 Simulated Annealing

The four important components required to use a simulated annealing process explained in section 2.2, are defined as:

- **System Description** The system configuration P 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 P randomly to configuration P', a random edge of the full edge list has its weight changed, therefore the minimum spanning tree that is calculated changes.

- **Quantitative Function** To evaluate the system configuration $E(P)$, the sum of the overlapping areas is used.
- **Annealing Schedule** The temperature T 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 number of iterations.

After reading in the mesh into the Polyhedron_3 data structure preparations are necessary to be able to apply simulated annealing. First as described in chapter 4, the algorithm calculates a dual graph and Gluetags, 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 100.000 and 1.0 per iteration, as experimental values, defining the run-time 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 pre-calculated Gluetags and chooses a Gluetag 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 face the Gluetag targets has a Gluetag yet

The algorithm discards all Gluetags 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 minimise the number of Gluetags to reduce time and effort on reconstruction.

After these steps, the algorithm starts unfolding the triangles and the Gluetags alike, as seen in figure 5.1. It calculates the area of triangle-triangle overlaps, Gluetag-triangle overlaps, and Gluetag-Gluetag 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 Gluetag related overlaps. This step is used to optimise the calculation time as well as to influence the evolving graph. This prioritisation is applied since a configuration that does not have any Gluetag 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 Gluetags overlap are left, these have a chance to be resolved for example by changing which Gluetags 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 minimise in this process. This initial configuration is saved as the best configuration P, as seen

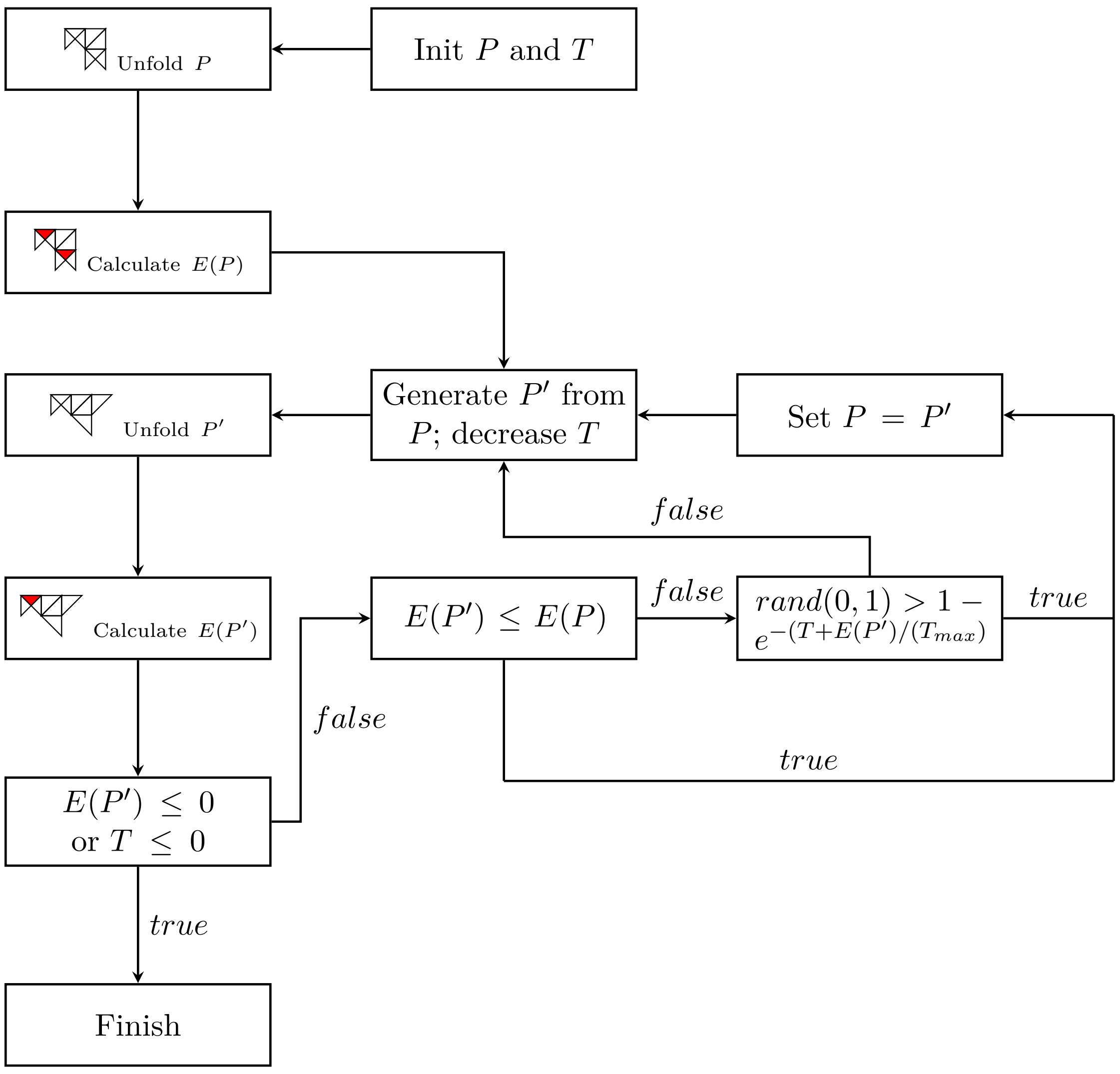


Figure 5.1: Overview of the simulated annealing process.

in figure 5.1. For each following iteration before these steps are repeated. A random edge will be chosen and assigned a new random weight, which possibly changes the next calculated minimum spanning tree and generates the configuration P'.

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 Gluetags 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 P' is smaller or equal to the energy of the previous configuration P, P will be set to P'. 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 visualised 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 in configuration P' or if the temperature reaches the minimum of 0, 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.

5.3 Parameterisation of the System

The algorithm is adjustable 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 Gluetag, the solution space is widened or made smaller if the Gluetags area increases. The default value is for it to be one fifth of the height of the face it targets.

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. The default number of iterations is 100.000.

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

CHAPTER 6

Results and Evaluation of the System

In this chapter experimental results are presented and their specifics are discussed. A quantitative performance analysis is also presented, analysing the time it takes to unfold various models. Furthermore, the limitations of the suggested approach are presented.

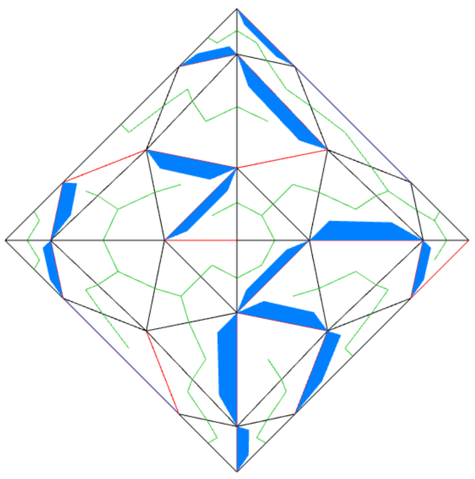
6.1 Results

This thesis proposes a simple spanning tree approach to find unfoldings of a mesh. A simulated annealing process is optimising the search for an overlap free unfolding by finding an optimal global layout.

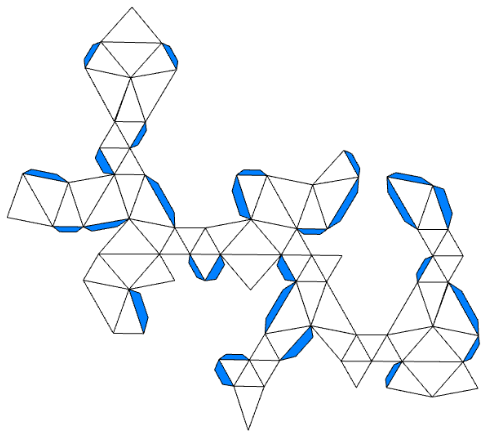
The following figures show the resulting unfolding of the implementation. The Gluetags are visualised in the 3D-Model and the unfolding, furthermore, the minimum spanning tree (green lines) and cut-edges (red lines) are visualised in the 3D-Model.

Figure 6.1 and 6.2 are sphere-like objects with a low count of faces, which is optimal for the proposed algorithm, as a solution can be found fast. For figure 6.1 three different unfoldings were generated, each having a different shape while it was reached in a different amount of time. This is due to the non-determinism of the algorithm, as well as the random walk when searching an unfolding.

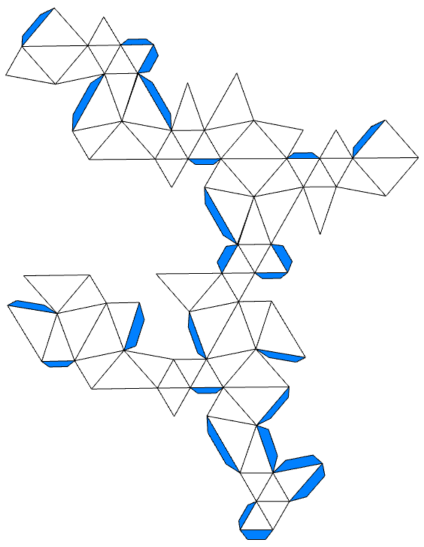
Contrary to the previous two figures figure 6.3 and 6.4 are star-like objects with 6.3 having a higher amount of faces. Both of them are harder to solve as they lose their spherical properties, which make unfolding easier. Figure 6.4 also shows the stretching of the unfolding that happens due to the structure of the model, which can also be observed in figure 6.7. It has very steep angles, which increases the likelihood to produce overlapping faces when unfolded more compact. This makes it harder to compute an unfolding, as the random walk takes more time to stretch out the unfolding.



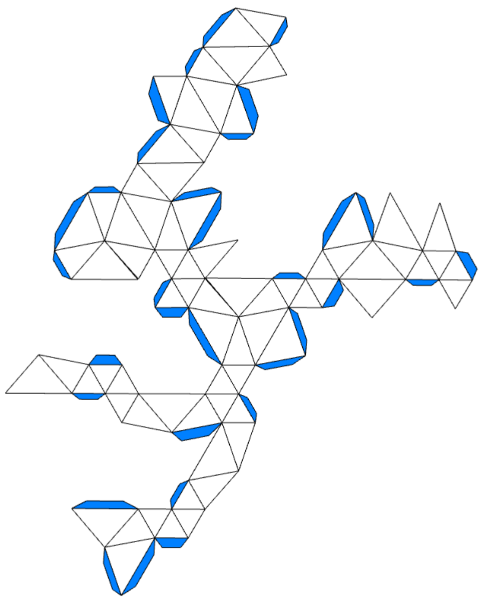
(a) 3D model of a star.



(b) Unfolding of the model in 31 seconds.



(c) Unfolding of the model in 18 seconds.



(d) Unfolding of the model in 77 seconds.

Figure 6.1: 3D Model with 72 faces and three possible unfoldings.

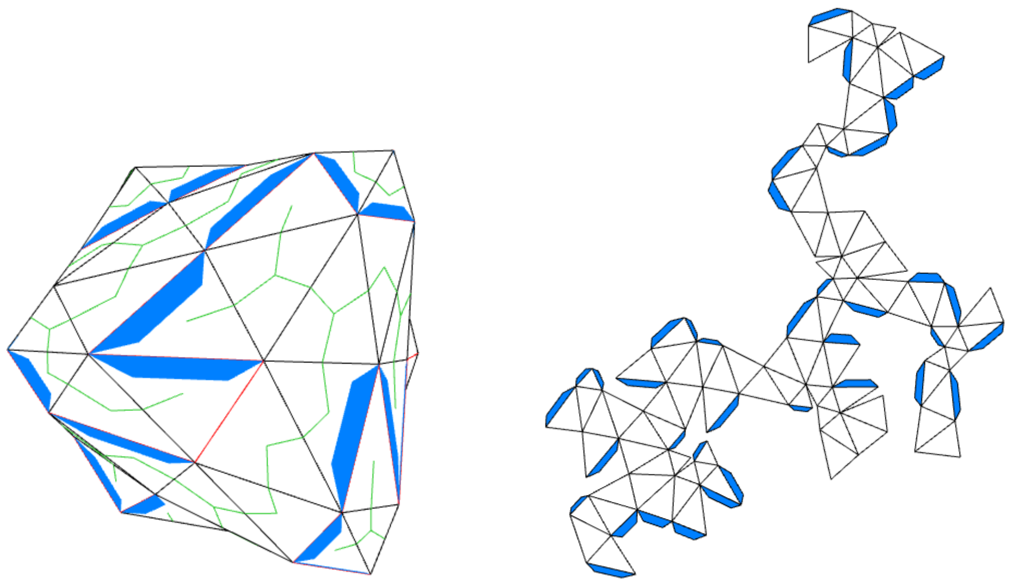


Figure 6.2: 3D Model of a star with 96 faces and the corresponding unfolding.

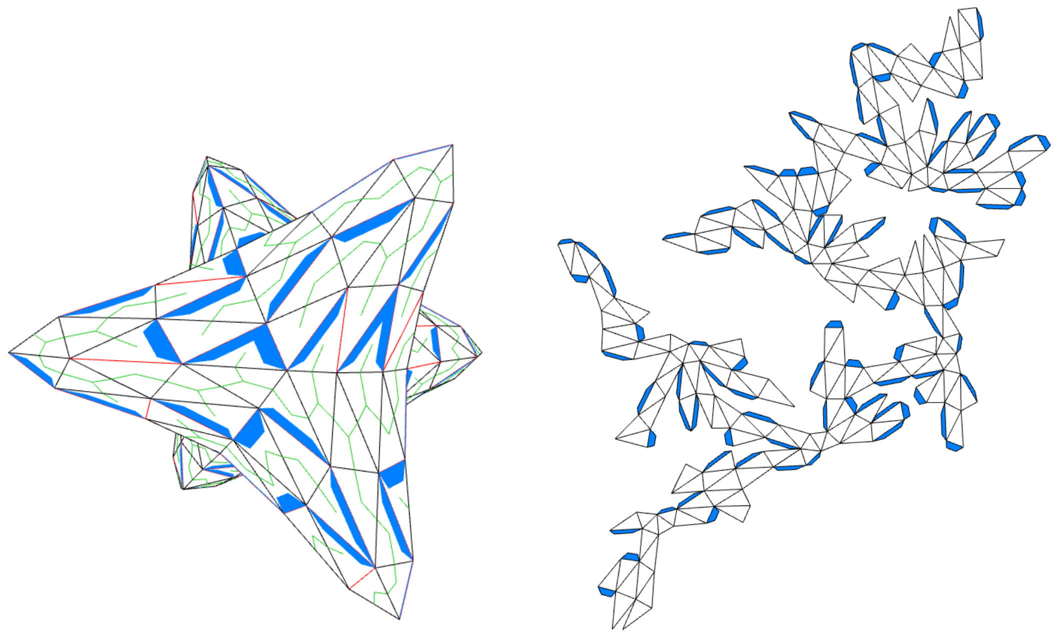


Figure 6.3: 3D Model of a star with 216 faces and the corresponding unfolding.

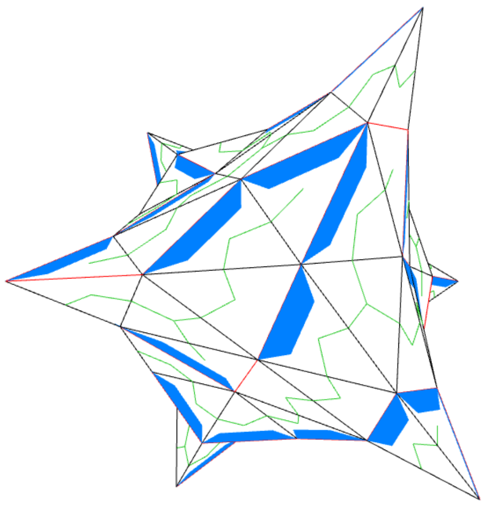
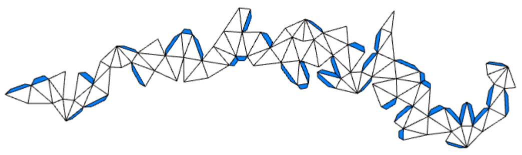



Figure 6.4: 3D Model of a star with 96 faces and the corresponding unfolding.

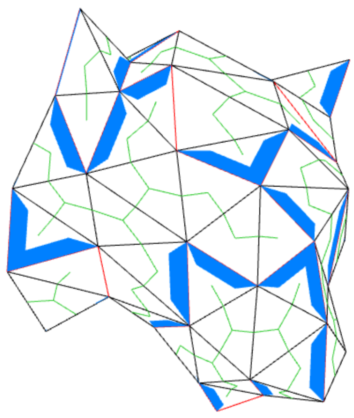
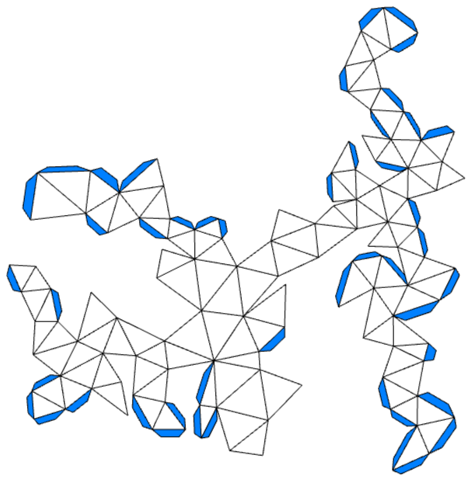



Figure 6.5: 3D Model of a tiger head with 112 faces and the corresponding unfolding.

Figures 6.5 to 6.11 are organic-like meshes with an increasing face count. The average time to find an unfolding increases, as can be seen in table 6.1, but still results are produced. Most of the higher face-count models produce stretched-out unfoldings, as the Gluetags make compact solutions less likely due to their need for space between faces.

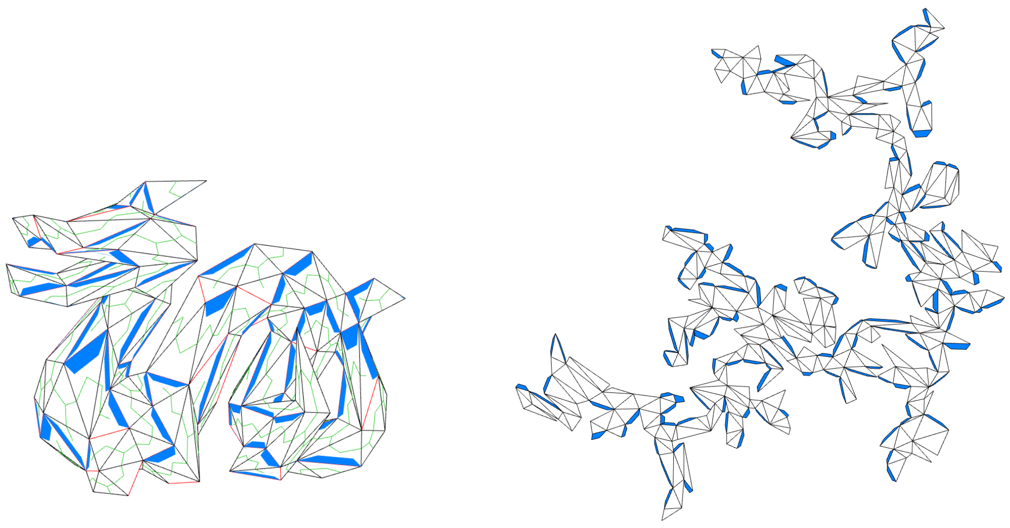


Figure 6.6: 3D Model of a dragon with 344 faces and the corresponding unfolding.

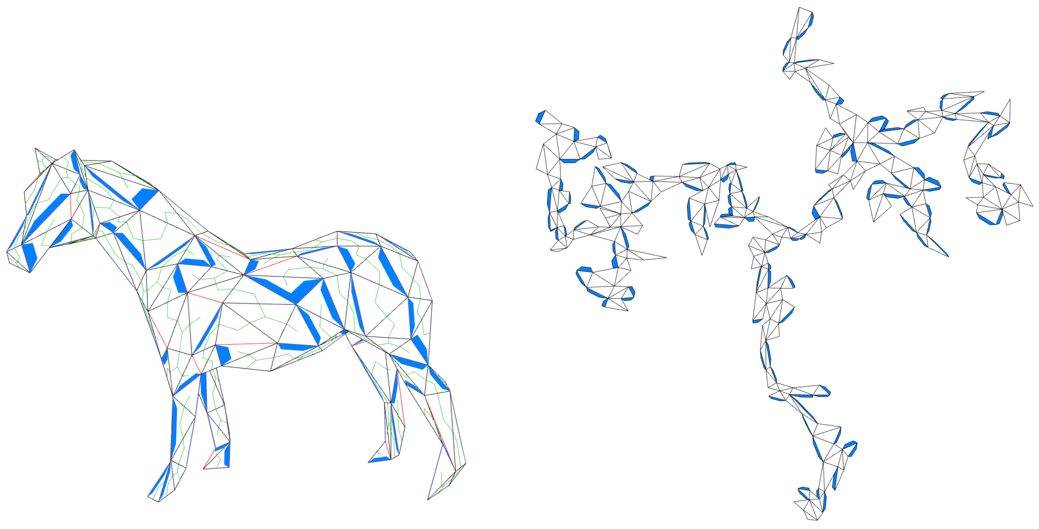


Figure 6.7: 3D Model of a horse with 312 faces and the corresponding unfolding.

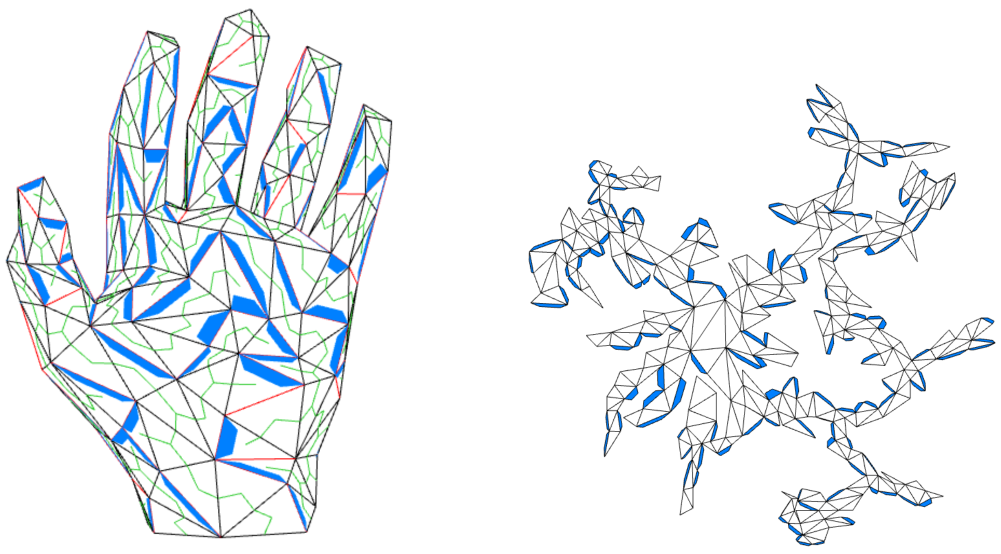


Figure 6.8: 3D Model of a hand with 336 faces and the corresponding unfolding.

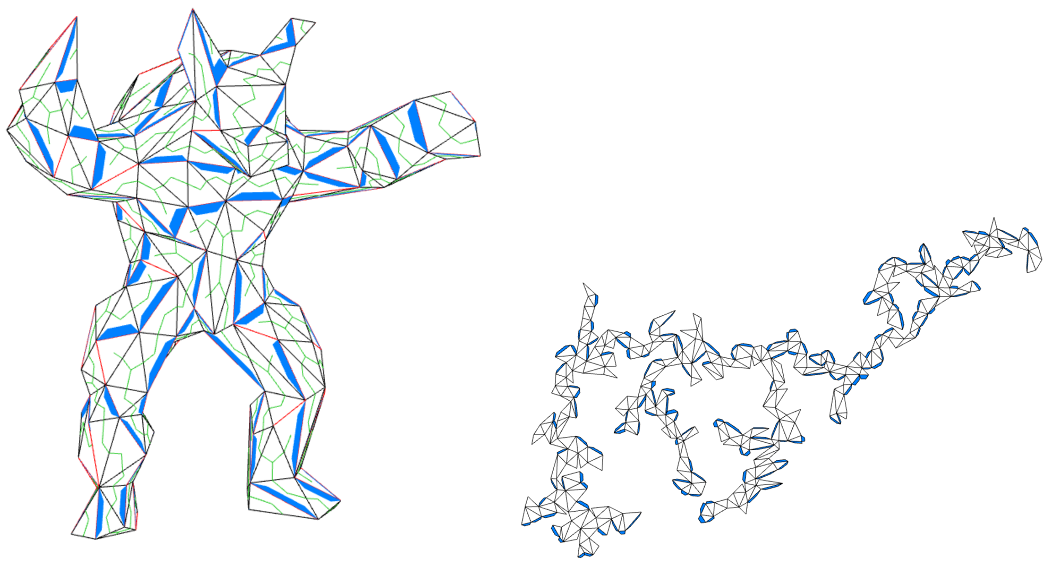


Figure 6.9: 3D Model of an armadillo with 386 faces and the corresponding unfolding.

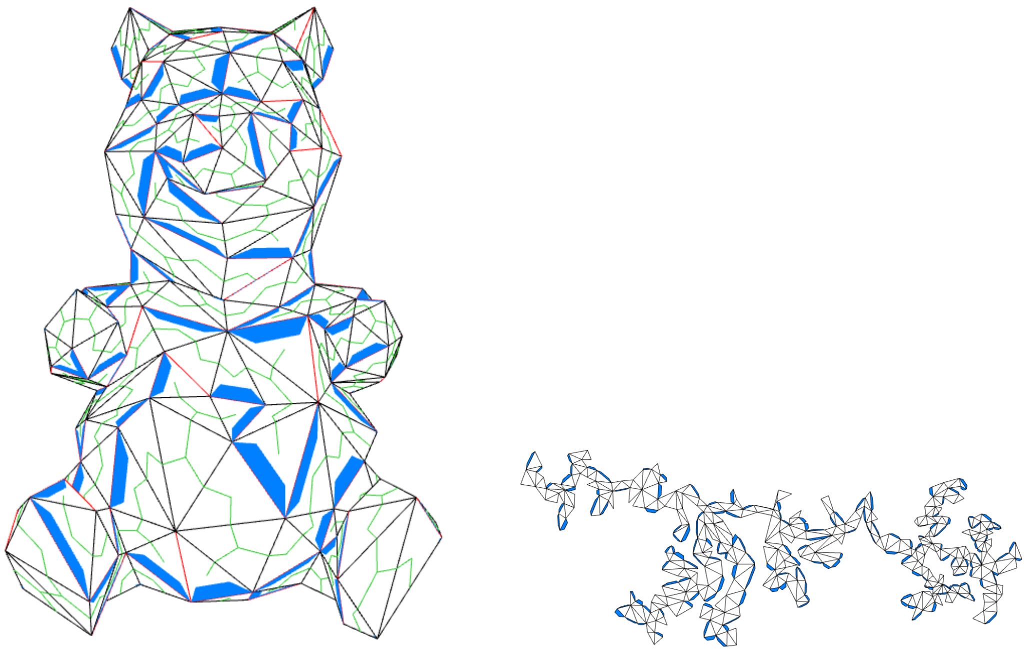


Figure 6.10: 3D Model of Winnie Pooh with 392 faces and the corresponding unfolding.

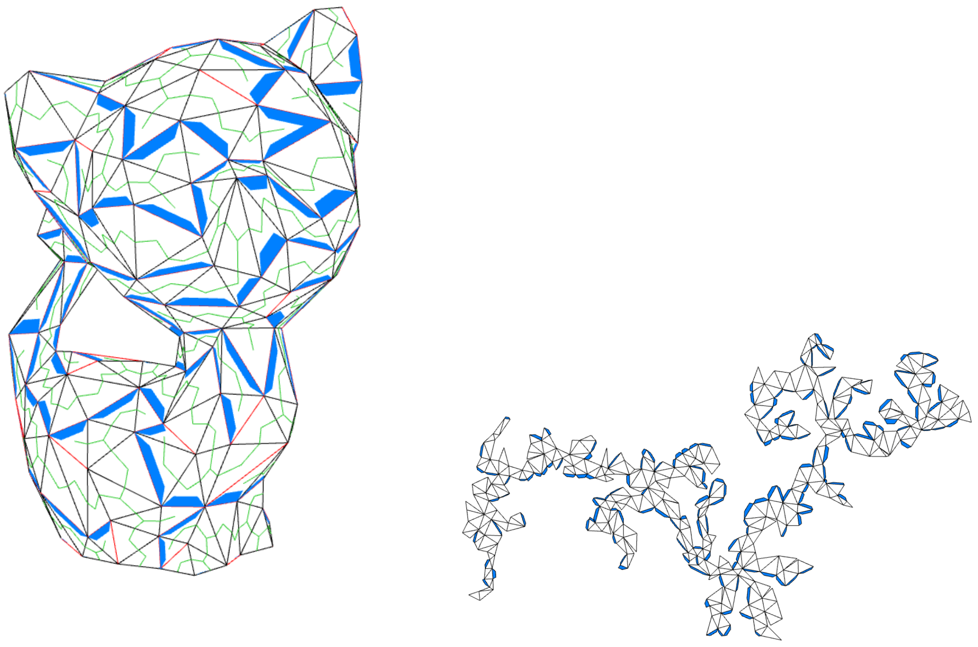


Figure 6.11: 3D Model of an cat with 392 faces and the corresponding unfolding.

6.2 Performance

To evaluate the algorithm and its implementation experiments were run with a variety of models. The results in table 6.1 were observed using a Gluetag 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 Gluetag. The default parameters have been used for these experiments.

Model		Vertices	Faces	Edges	Time to Unfold (s)	Time to Bruteforce (s)
Octa		6	8	12	0	0
Icosa		12	20	30	0	0
Star		14	24	36	8	19
Star-Sqrt3	(Fig. 6.1)	38	72	108	31	>60000
Star-4Split		50	96	144	435	-
Star-Loop	(Fig. 6.2)	50	96	144	137	-
Star-Butterfly	(Fig. 6.4)	50	96	144	1047	-
Tiger	(Fig. 6.5)	58	112	168	65	-
Kitten		64	122	184	48	-
Moneybox-128		64	128	190	160	-
Bunny-128		66	128	192	103	-
Moneybox-196		98	196	292	324	-
Star-PNsplit	(Fig. 6.3)	110	216	324	625	-
Snail-286		145	286	429	1315	-
Horse	(Fig. 6.7)	152	302	452	946	-
Hand	(Fig. 6.8)	170	336	504	1377	-
Dragon	(Fig. 6.6)	172	344	514	1292	-
Bunny-348		176	348	522	976	-
Luigi		180	356	534	686	-
Armadillo	(Fig. 6.9)	195	386	579	730	-
Pooh	(Fig. 6.10)	198	392	588	957	-
Moneybox-392	(Fig. 6.11)	196	392	586	2200	-
Meister	(Fig. 6.12)	200	394	592	3900	-
Gear		256	508	762	-	-
Cat		353	702	1053	-	-
Fish		477	950	1425	-	-
Mannequin		690	1376	2064	-	-

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 Gluetags. The bigger the Gluetags 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

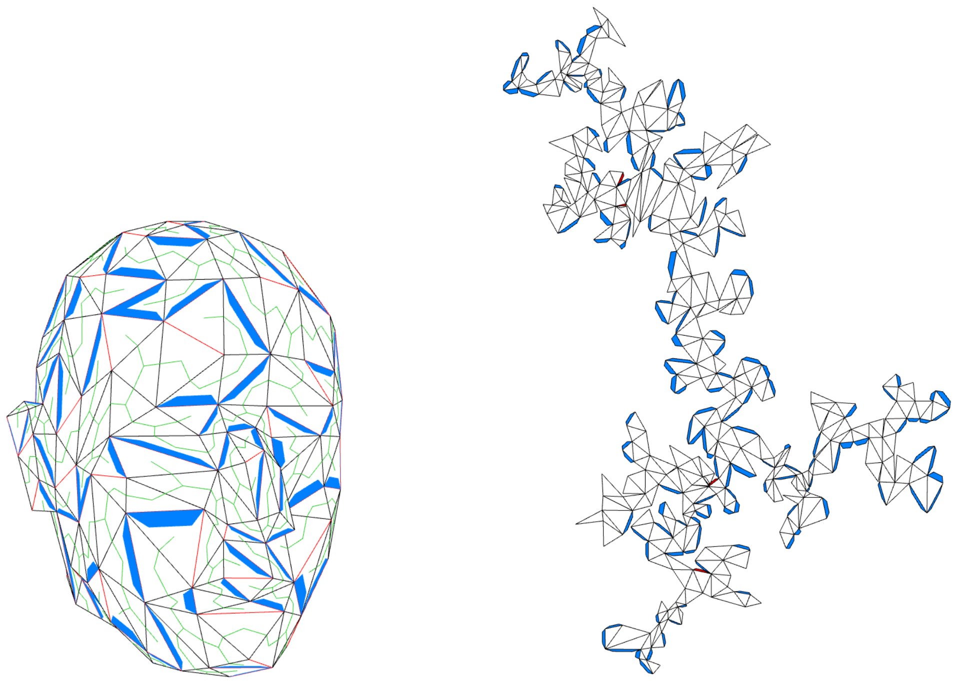


Figure 6.12: 3D Model of the Meister with 400 faces with the failed unfolding after 100000 iterations. Overlapping Gluetags are coloured red.

and which edges are changed. For models marked with a - in the table 6.1 no unfolding was found within 100000 iterations. Using a bruteforce algorithm solutions for small models is possible, but as the number of faces grows the calculation using bruteforce is not feasible anymore.

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 Gluetags that are necessary cannot be calculated beforehand. Neither the amount nor the position of the Gluetags 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. This can be seen in the performance table 6.1, where even for small models the amount of time necessary for a brute force algorithm to find a solution is too high even for small models.

Another limitation that does not have only negative consequences is the random-walk

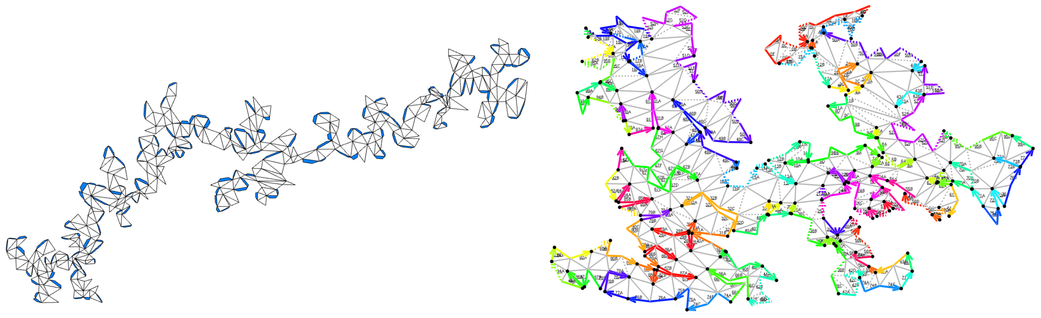


Figure 6.13: (left) Unfolding of the proposed approach. (right) Unfolding with the algorithm proposed by Takahashi et al.[TWS+11]

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. Figure 6.12 shows an unfolding proposed after 100,000 iterations that failed to generate an overlap-free solution. When the iteration cap was lifted, an unfolding could be found, as seen in figure 6.13, after around 130,000 iterations. The figure also shows a comparison between the approach suggested in this thesis and the algorithm proposed by Takahashi et al.[TWS+11]. The unfolding of their approach can be more since no Gluetags are restricting the space used.

CHAPTER 7

Conclusion

In this chapter a short summary of the results is provided and future work is presented.

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 Gluetags 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 this factor can be disregarded as it can be added using this approach by 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 quality using the overlap area but also considering other factors, like maintaining structures of the mesh together to assist users in the reconstruction of the models.

Another possible improvement would be to ensure efficient use of paper space. This mainly requires the avoidance of stretched out unfoldings as seen in Figure 6.4. To achieve this, the quality metric of the unfolding can be evaluated not only using the overlapping area but also a value that describes how stretched out the unfolding is.

Performance can be improved, if Gluetags 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. To compute necessary modifications to the Gluetag the intersection points of the Gluetag and the triangle can be taken as new vertices of the Gluetag.

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

To avoid manual adjustment of the maximum number of iterations required to find unfoldings for large models, the number of required iterations might be calculated using an exponential function using the number of faces of the mesh.

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.

[CY17] Yi-Jun Chang and Hsu-Chun Yen. Improved algorithms for grid-unfolding orthogonal polyhedra. *International Journal of Computational Geometry & Applications*, 27(01n02):33–56, 2017.

[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.

[DT96] Ronald A DeVore and Vladimir N Temlyakov. Some remarks on greedy algorithms. *Advances in computational Mathematics*, 5(1):173–187, 1996.

[FTS$^+$13] Samuel M Felton, Michael T Tolley, ByungHyun Shin, Cagdas D Onal, Erik D Demaine, Daniela Rus, and Robert J Wood. Self-folding with shape memory composites. *Soft Matter*, 9(32):7688–7694, 2013.

[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.

[HE12] Thomas Haenselmann and Wolfgang Effelsberg. Optimal strategies for creating paper models from 3d objects. *Multimedia systems*, 18(6):519–532, 2012.

[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.

[LES08] Kwang Y Lee and Mohamed A El-Sharkawi. *Modern heuristic optimization techniques: theory and applications to power systems*, volume 39. John Wiley & Sons, 2008.

[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.

[Tib14] Skylar Tibbits. 4d printing: multi-material shape change. *Architectural Design*, 84(1):116–121, 2014.

[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.

[Wik19] Wikipedia contributors. Sutherland–hodgman algorithm, 2019. [Online; accessed 01-August-2019].

[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.