Optimising 3D Mesh Unfoldings with Additional Gluetabs using Simulated Annealing

Thorsten Korpitsch*
Supervised by: PhD Hsiang-Yun Wu†

Institute of Visual Computing & Human-Centered Technology
TU Wien
Wien / Austria

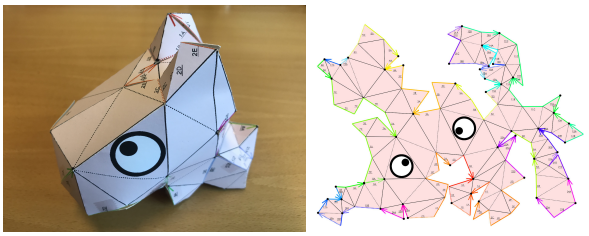
Abstract

3D Mesh unfolding is a process of transforming a 3D mesh into one or several 2D planar patches. The technique is widely used to produce papercraft models, where 3D objects can be reconstructed from printed paper or paper-like materials. Nonetheless, the reconstruction of such models can be arduous. In this paper, we aim to unfold a 3D mesh into a single 2D patch and introduce Gluetabs as additional indicators and in order to give users extra space to apply glue for better reconstruction quality. To avoid unnecessary Gluetabs, we reduce their number, while still guaranteeing the stability of the constructed model. To achieve this, a minimum spanning tree (MST) is used to describe possible unfoldings, whereas simulated annealing optimisation is used to find an optimal unfolding without overlaps. We aim to unfold 3D triangular meshes into single 2D patches without applying shape distortions, while appropriately assigning a reasonable amount of Gluetabs. Moreover, we incorporate a visual indicator scheme as a post-process to guide users during the model reconstruction process. Our quantitative evaluation suggests that the proposed approach produces fast results for meshes under 400 faces.

Keywords: Mesh unfolding, Simulated annealing, Gluetabs

1 Introduction

Papercraft is a popular art, where people create 2D or 3D objects from cardboard or paper, as shown in Figure 1. To achieve this, a 3D mesh representing the object needs to be unfolded into a single- or multiple 2D patches, which can then be printed and used to reconstruct the model in 3D. Models, that can be created, range from simple ones, such as paper aeroplanes, to complex models, for example of buildings. Recently, it is also used in combination with self-folding materials to form structures in an automatic fashion [12]. As demonstrated by Takahashi et al. [23], unfolding a 3D triangular mesh into a single patch, in contrast to multiple patches, eases the reconstruction process for users. However, finding such an unfolding, in which no pair of faces overlaps with one another and without distorting the original mesh is a difficult task. More specifically, it has been proven as an NP-complete problem [11]. Moreover, the reconstruction of a model can also be hard, even with indicators that show which cut-edges should be glued together, as presented by Takahashi et al. [23].



(a) (b)

Figure 1: An example of a papercraft model, which includes (a) a cow 3D model and its corresponding (b) single unfolded patch.

In this paper, we propose to add a minimal number of *Gluetabs* to ease the reconstruction of models by guiding which cut-edges should be glued together as well as providing users sufficient space to apply glue. The addition of Gluetabs increases the complexity of finding an overlap-free unfolding, due to the combinatorial complexity of selecting cut-edges to apply Gluetabs.

Gluetabs are essential as they allow users to build clean 3D models and guide users during reconstruction process. This is achieved by calculating minimum spanning trees of the dual graph of the 3D mesh which describes a possible unfolding. Simulated annealing optimisation is used to find an overlap-free unfolding. Gluetabs are pre-calculated and treated analogue to mesh-faces when unfolding the model.

Our experiments suggest that models with less than 700 faces can be unfolded using the proposed approach effectively, whereas an increasing number of faces requires more time to find appropriate unfoldings. For meshes with

*e01529243@student.tuwien.ac.at
†wu@cg.tuwien.ac.at

more 700 faces, no unfolding can be found within a reasonable timeframe. Another key factor that influences the unfolded results is the number and size of the Gluetabs, as well as the integration with the unfolded patches. We select a trapezoid shape as the design of a Gluetab. The advantage of using the trapezoid shape is that a trapezoid consists of two triangles, so that we can consider Gluetabs as additional faces of the 3D mesh and apply similar overlap-detection when searching a feasible solution. Furthermore, this shape gives users more space than a simple triangle. It is also preferable to a rectangle as it takes less space, because as the Gluetabs size increases, the difficulty of finding an overlap-free unfolding also increases. Shortly speaking, our contribution in this paper includes a new meta-heuristic approach of finding unfoldings of 3D meshes as well as introducing a minimum number of Gluetabs giving users space to apply glue to and guide reconstruction.

The remainder of this paper is structured as follows. Section 2 discusses previous explorations for mesh unfolding and optimisation techniques. Section 3 describes the concepts used in this paper. Section 4 gives an overview of the processing pipeline, that computes an unfolding and describes necessary steps in detail. Section 5 brings insight into the implementation of the previously explained approach and focuses on the simulated annealing process. Section 6 shows the results of the implemented approach and evaluates its performance and limitations. Finally, Section 7 summarises the findings and provides an outlook on future work.

2 Related Work

This section focuses on previous work done on the topic of optimising the unfolding of 3D models and also explains the differences in the approach proposed in this paper.

2.1 Optimised Unfolding of 3D Meshes

Mesh Unfolding has different applications, such as creating papercraft models [23, 20] and the creation of models from self-folding materials [8, 24]. Some techniques allowing mesh deformation [2, 16], in order to relax the problem. Mitani et al. [16] and Chang et al. [2] propose methods that allow mesh deformations when unfolding. Also, the theoretical perspectives [18] of the problem have also been explored. Moreover, some other authors study different types of target meshes, for example, orthogonal polyhedra [25, 5, 4]. The most relevant work of this paper is done by Takahashi et al. [23]. The authors proposed a genetic-based algorithm to find a single connected patch for printing purpose. They unfold 3D models using a heuristic approach and they do not distort or edit the original 3D model. The key concept of unfoldability is borrowed from topological surgery, in order to guarantee an unfolded patch can be stitch together from the corresponding cut edges. Our paper, on the other hand, explores a meta-heuristic simulated annealing approach to find unfoldings, in contrary to the work done by Takahashi et al. [23]. Simulated annealing has the advantage of its easiness to implement compared to genetic algorithms and it is more likely to find an optimal solution compared to a greedy algorithm [20].

Another key conventional approach is investigated by Straub et al. [20]. They explored the unfolding and Gluetabs on each cut-edge of the unfolding. They also explored the removal of overlaps by introducing new subdivisions to the mesh. They use a greedy algorithm to optimise the unfolding and to resolve overlaps. Gluetabs that have been added to an unfolding are optimised, i.e. changed in size to avoid overlaps, after an initial unfolding is found. To be able to print the unfolding is then separated on multiple cut-out sheets if it does not fit on a single one. In this paper, the proposed algorithm examines all possible Gluetabs in advance and selects a minimum number of Gluetabs at each unfolding iteration. To the best of our knowledge, the integration of unfolding and Gluetabs has not been explored in the state-of-the-art literature. Readers can consider this technique as an improvement of the approach studied by Takahashi et al. [23], where they found that it helps users with reconstructing a 3D model if the mesh is unfolded into a single patch.

2.2 Optimisation Techniques

Since the 3D mesh unfolding problem is considered an NP-complete problem [11], optimisation techniques are often used to find a solution. This is because trying all combinations is not practical in most of the cases. Many optimisation techniques are well explored, including greedy algorithms [7] or heuristic optimisation techniques [15]. This paper proposes using simulated annealing as the optimisation technique for mesh unfolding problem to find an optimal solution based on the cost function, because it is easy to code and has advantages over greedy algorithms. Simulated annealing is a well-known optimisation technique [13] that is widely applicable to problems found in computer science [9] [6] [1] and other scientific fields [17] [22].

Algorithm 1 depicts the concept of a simulated annealing approach, where it optimises a configuration P by minimising the energy of P. At each iteration, a new configuration P' is created, and the energy is compared to the previous configuration using the function $E(*)$. At each iteration, there is a chance to take a worse configuration, to avoid getting stuck in local minima, as shown at line 6 in Algorithm 1. k_B is a constant, which should be adjusted and determined through the result of experiments.

Since we could encounter local minima when optimising 3D Mesh unfolding, simulated annealing has a distinct advantage over greedy algorithms, as it is less likely to get stuck in a local minimum. Compared to genetic algorithms, simulated annealing is easier to code and works

Data: Configuration P, Max Temperature T_{max}
Result: Optimised Configuration P

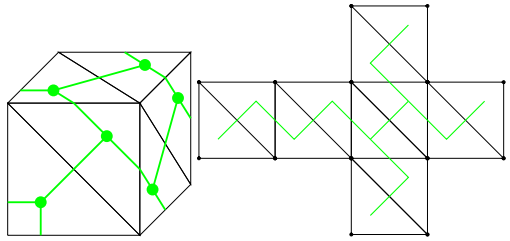
```
Set T = T_max;
while T > 0 do
    /* Random step to generate P'
       from P */
    Create P' from P;
    if E(P') ≤ E(P) then
        Set P = P';
    else if
     rand(0,1) ≤ exp(−(E(P') − E(P))/(k_B T))
     then
        Set P = P';
    Decrease T;
end
```

Algorithm 1: The pseudo-algorithm of simulated annealing.

well on problems with continuous cost functions. Therefore we chose simulated annealing as the optimisation approach.

3 Definition of Key Concepts

In this section the terminology used in this paper is defined. Further key concepts are described.



(a) (b)

Figure 2: (a) A triangular mesh of a box (black) and its dual graph (green). (b) A MST (green), of the dual graph, representing an 2D unfolding.

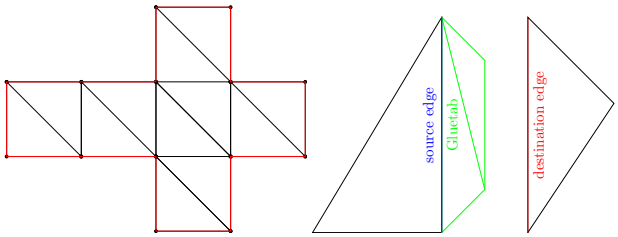
Triangular Mesh M. The input for the algorithm is a triangular mesh $M = (V_M, E_M, F_M)$. We restrict the meshes for this approach to genus zero.

Dual Graph D. $D = (V_D, E_D)$ is called the dual graph of M and is obtained by calculating a dual vertex V_D for each face $f \in F_M$ and a dual-edge E_D for every two faces separated by an edge $e \in E_M$ [10]. 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 computation of it can be done very efficiently, which makes it a very compelling data structure to use for 3D mesh unfolding.

Minimum Spanning Tree (MST) T. Given a cost-function $c(v,w)$ for each edge $(v,w) \in E_D$, a minimum spanning tree $T = (V, E')$ can be calculated such that $\sum_{\{v,w\} \in E'} c(v,w)$ is minimal [3].

Therefore the MST in combination with the dual graph is a good tool to calculate possible unfoldings. In order to calculate a MST, we need to assign a weight to each edge in the dual graph. To achieve this, we propose to assign a random weight between $(0,1)$**??**

2D Unfolding. A 2D unfolding is defined as the 2D representation of the 3D Model, after being unfolded. It is computed by unfolding faces one after another, referring to the MST T. The exact order of which face is unfolded first can be neglected as the MST defines exactly one unfolding. In this paper, we define the quality of an unfolding by the summed area of overlaps. A correct unfolding has, therefore, an overlapping area of 0. A single 2D Unfolding describes the unfolding of a 3D mesh into one connected planar patch.



(a) (b)

Figure 3: (a) Cut-Edges (red) and Bend-Edges (black). (b) Exemplary trapezoid Gluetab (green) attached to its source face edge (blue) and the face it will be glued to (red).

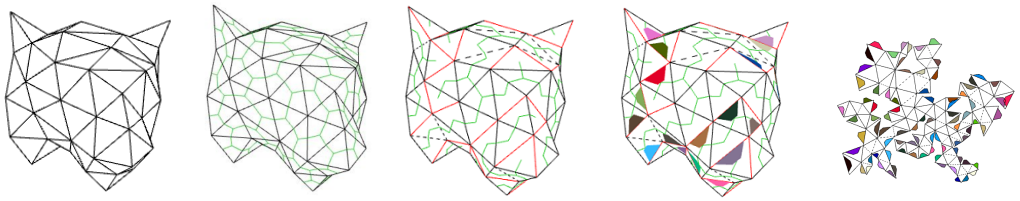
Cut-Edge. Each edge that is a boundary edge in the unfolding is referred to as a Cut-Edge. Each face has at least one Cut-Edge. Gluetabs might be attached to a Cut-Edge as shown in Figure 3(a).

Bend-Edge. Every edge in T is referred as a Bend-Edge, as shown in Figure 2(b) and Figure 3(b), therefore complementing the Cut-Edges, with each face having at least one Bend-Edge.

Gluetab. A Gluetab refers to an additional space that users use to apply glue for attaching Cut-Edges to each other. It can have different forms, for example, the shape of a trapezoid, as shown in Figure 3(b). For each Gluetab, it contains one source and one destination edge. In this paper, we propose trapezoid-shape Gluetabs as an experimental shape. This is because trapezoid is a standard shape for Gluetabs, and of course this can be extended to other preferred structures.

4 Unfolding Meshes with Gluetabs

Figure 4 gives a step-by-step overview of the present algorithm. It shows in total five steps. This includes computing the dual graph (S1), pre-computing Gluetabs (S2),



(a) Original Mesh (b) Dual graph (c) MST (d) Gluetabs Selected (e) Unfolding

Figure 4: Steps of the proposed unfolding process. (a) Shows the 3D Model. (b) Shows the dual graph. (b) Shows a MST. (c) Shows a minimal number of selected Gluetabs. (d) Shows the unfolded patch with visual markers.

computing the MST (S3), selecting an appropriate number of Gluetabs (S4) and then unfolding with additional visual markers (S5).

After loading a mesh, as seen in Figure 4(a), the corresponding dual graph is computed, as highlighted in green in Figure 4(b). At this step, for each edge, Gluetabs are pre-computed for each side of each edge. Once the dual graph is calculated a MST is computed (highlighted in green in Figure 4(c)), whereas the resulting cut-edges are highlighted red. Based on the MST T, the Gluetabs are selected, as explained in section 4.4 (Figure 4(d)). The next step is to unfold the model and Gluetabs by referring to the MST T (Figure 4(e)). If the unfolded faces are overlap, we change the weight of a random edge and return to S2. Otherwise, if no overlaps occur, a post-processing is conducted as described in Section 4.7, which results in an unfolded patch, as seen in Figure 4(d).

4.1 Dualgraph Generation (S1)

The dual graph is calculated once at the beginning, as the original mesh is not changed, as shown in Figure 4. The neighbourhood relation of the faces can be derived from edges connecting the two vertices the faces have in common. For this reason, we iterate through all faces of the mesh in two nested loops and save a dual edge each time we find a new face with common vertices with another face. In order to compute a MST, each new dual edge is assigned a random weight between $(0, 1)$.

4.2 Gluetabs Introduction (S2)

The second step is to calculate Gluetabs (see Figure 4(b)). For each edge in the dual graph D, a Gluetab is calculated for both facets connected by this edge. The endpoints of an edge are the base of a Gluetab, as shown in Figure 3(a). The algorithm that is applied to faces of the mesh is also applied to the Gluetabs. The size of Gluetabs are pre-computed, and the height of a Gluetab takes maximum 20% of the target face. The shape and size of the Gluetabs are determined experimentally to provide users with appropriate extra space. Gluetabs are calculated once after the dual graph was calculated, since the dual graph does not change.

4.3 Generation of MST (S3)

The algorithm then computes a MST from the dual graph. This is done using Kruskal's algorithm [14] to find the shortest spanning subtree on the dual graph D. The resulting aggregation E' contains all Bend-Edges and its complement contains all Cut-Edges. Using E', the mesh can be unfolded, whereas its complement is used to determine which Gluetabs are necessary to reconstruct a stable model.

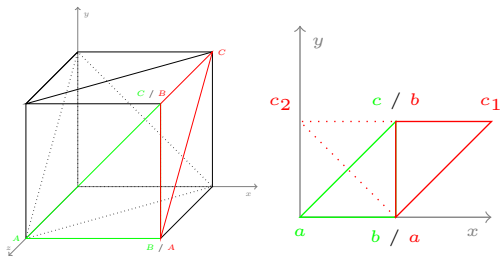
4.4 Gluetabs Selection (S4)

Based on the MST, the Gluetabs that are part of the unfolding can be selected. By iterating through all possible Gluetabs, a Gluetab is selected if at least one of the following conditions is fulfilled:

- No Gluetab for this face was selected.
- The number of successive Cut-Edges no Gluetab was selected is higher than 1.

With these conditions on reconstruction, the built model has no moving vertices. In other words, for each vertex, there is maximum one adjacent edge is not fixed using a Gluetab, which yields a stable 3D model. The stability is tolerable here because adding more Gluetabs will not yield a more stable reconstructed model while increasing the reconstruction effort.

4.5 Unfolding the 3D Mesh using MST (S5)



(a) (b)

Figure 5: (a) A 3D model of a box, and (b) the unfolding of the first two faces.

Figure 4 shows how an unfolding of the mesh is calculated Followed by the calculation of the MST. By picking an edge in E, we can then decide the two faces to unfold. Faces can be unfolded by setting two vertices of the first triangle and then calculating the unknown vertices for each triangle, as shown in Figure 5 or using transformation matrices. This step is computed iterative until all faces and Gluetabs are unfolded.

4.6 Detecting Overlaps

The last step of the simulated annealing loop, is to determine the quality of the unfolding by calculating its energy function $E(P')$. The energy function $E(P')$ is computed by collecting the total pixels of the overlapping region, to guide the algorithm for finding an overlap-free single patch with Gluetabs.

An overlap is defined if two faces are intersected with each other partly (see Figure 6), or if a face is entirely located inside another Face. For Gluetabs, the definition is similar. In total, three different cases of overlaps can occur, including either face-face overlaps, Gluetab-Gluetab overlaps, or face-Gluetab overlaps.

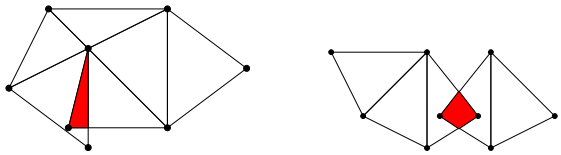


Figure 6: Overlaps of faces resulting in different overlapping areas.

To improve the performance of finding overlap-free unfoldings, face-face overlaps are checked before the Gluetabs are checked. This is because if faces overlap with each other, an overlap-free unfolding cannot be found even if the Gluetabs are overlap-free. Moreover, the performance can be improved by limiting the number of faces that need to be checked. As the overlap detection does not need to be done with its predecessor, because they share a common edge an overlap is not possible.

If an overlap occurs, the Sutherland-Hodgman Clipping algorithm [21] is used to calculate the overlapped area. This algorithm finds the vertices of the intersection polygon created by the two faces. The overlap can be described as a polygon (see Figure 6). The area is calculated with the shoelace formula $Area = \frac{1}{2}\left|\sum_{i=0}^{P-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right|$ [19], where x_i and y_i are the coordinates of the i-th point in P.

4.7 Post-Processing

Two strategies have been introduced to improve the visual quality of the unfolded results.

Colour Coding. To support reconstructing and edge-Gluetab mapping, the Gluetab is coloured, and the Gluetab is also mirrored on the targeted edge. Different colours are chosen for each Gluetab to help identify where a Gluetab needs to be glued to.

Bend-Edge Coding. To make it easier in which direction a Bend-Edge needs to be bent, each Bend-Edge is coded to distinguish between a mountain-fold or a valley-fold, as proposed by Takahashi et al. [23]. A mountain-fold is represented by a solid line and the valley-fold is displayed by a dotted line. To identify the fold-type, the normal vector n_1 of the first face is calculated, and a vector e_2 of the common edge to the not-shared vertex of the second face is calculated. Then the dot product of n_1 and e_2 is negative if it is a valley-fold and positive if it is a mountain-fold, which can easily be seen in the 3D and 2D representation in Figure 4(d).

4.8 Parameters used in the Approach

The algorithm is adjustable by changing multiple parameters without changing the approach itself. By configuring the temperature T_{max} can be adjusted to define the number of iterations the algorithm runs, the default value is 100.000. k_B can be adjusted as well, its default value is 0.001.

5 Implementation Details

We define the configuration P as the list of weighted dual graph edges, from which the MST can be derived. To generate a new configuration P' from P, the weight of a random edge is randomly changed. The quantitative evaluation function $E(P)$ is defined as the sum of overlapping areas of an unfolding.

Our data structure holds information about the neighbourhood relations between vertices and edges. It also holds information about all faces. After loading in the mesh into a data structure, preparations are necessary in order to apply simulated annealing. The algorithm calculates a dual graph and Gluetabs, for each edge of the dual graph, as each edge could be a possible cut edge. Furthermore, we set the maximum temperature to $100,000$ and the cooling rate 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. This is summarised as (1) in Figure 7.

Then it calculates an initial MST using the edge list. Gluetabs are calculated for all edges that are cut, which are the edges not present in the MST. The algorithm iterates through all pre-computed Gluetabs and selects Gluetabs if they fulfil the conditions from Section 4.4. After these steps, the algorithm starts unfolding the faces and the Gluetabs alike ((2) in Figure 7).

The algorithm then computes the area of face-face overlaps, face-Gluetab overlaps, and Gluetab-Gluetab overlaps, described as (3) in Figure 7. This approach proposes to weigh face-face overlaps with a higher factor

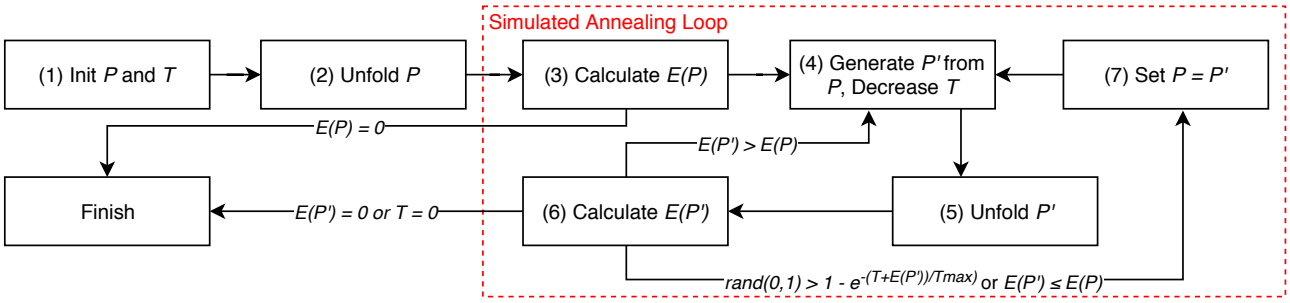


Figure 7: Overview of the simulated annealing process.

than Gluetab-Gluetab overlaps. This prioritisation is applied since a configuration that does not have any Gluetab overlaps, but still has face overlaps is less optimal than the other way around. If no face-face overlaps occur in a configuration and only Gluetabs overlap are left, these have a chance to be resolved for example by changing which Gluetabs are used or by post-processing the size of the Gluetabs.

If $E(P)$ is zero the algorithm terminates, otherwise we generate a new configuration P' from P by changing the weight of a random edge to a random value and decrease the temperature T, as seen in (4) in Figure 7.

P' is then unfolded again ((5) in Figure 7). If the energy of the new configuration P', (6) in Figure 7, is smaller or equal to the energy of the previous configuration P, P is set to P', (7) in Figure 7. 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 decreases the likelihood of the algorithm getting stuck in a local minimum.

At the end of every iteration, if P is set to P', it is visualised using OpenGL. The annealing process terminates if $E(P')$ reaches 0 or if T reaches the minimum of 0, which means that it ends without finding an unfolding.

6 Results and Evaluation

This paper proposes a spanning-tree based approach to find a single unfolding 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 results generated using our system. The Gluetabs are highlighted in the original 3D mesh. Furthermore, the MST (green) and cut-edges (red) are visualised in the 3D mesh. Note that the specification of testing data is summarised in Table 1.

The system is implemented on a laptop with an Intel Core i7 CPU (4 cores @ 3.3 GHz, 4MB L3 Cache) and 8GB RAM. The source code is written in C++17. The OpenGL ver. 4.5 library is used for the visualisation of the algorithms step as well as the results. The CGAL ver. 4.13 library is used to read in off files and to provide the underlying data structure. The Graphical User Interface is developed using the Qt Library ver. 5.13, CMake ver. 3.14 is used to manage the build process, and sources were compiled with GCC 9.2 on Ubuntu 18.04.02 LTS. Temperature T is set to 100,000 with a cooling rate of 1.0, resulting in a maximum of 100,000 iterations.

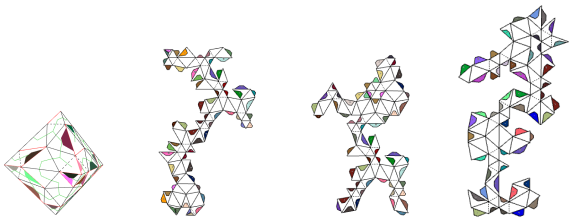


Figure 8: A 3D sphere-like objects with 72 faces and three of possible unfoldings.

Figure 8 is a sphere-like objects with a low count of faces , which work best with the proposed algorithm, as a solution is found fast. In Figure 8, three different unfoldings are presented. Each having a different shape and taking different amounts of time, 18, 77 and 49 seconds. This is due to the non-deterministic property of the algorithm, as well as the random walk when searching a feasible unfolding.

Figures 9a, 9d, 9f, 9g and 9h show meshes that have partly thin parts that lead to a further stretch of the unfolding even after optimising the spatial use of the unfoldings.

Figures 9b, 9c and 9e are again more sphere-like which in general leads to more compact unfoldings, but can also produce stretched results due to the randomness of simulated annealing.

6.1 Quantitative Evaluation

To evaluate the algorithm, we conduct an experiment through running a variety of 3D models. The results are summarised in Table 1 using the previously described Gluetab properties. In the experiment, a maximum of 100,000 iterations is used limit the computational time. The time in Table 1 is the average time needed for computation for 2 runs, it does only include computation time, rendering the results is excluded. The bruteforce approach

(a) Dragon with 144 faces. (b) Star with 216 faces. (c) Snail with 286 faces. (d) Horse with 302 faces.

(e) Bunny with 328 faces. (f) Hand with 336 faces. (g) Armadillo with 386 faces. (h) Cat with 392 faces.

Figure 9: Various 3D models (left) and their corresponding unfoldings (right).

tries all permutations of MST to find a solution and returns the first solution found.

Model		Vertices	Faces	Edges	Time (s) Our Approach	Bruteforce
Octa		6	8	12	<1	<1
Icosa		12	20	30	<1	<1
Star		14	24	36	8	19
Star-Sqrt3	(Fig. 8)	38	72	108	31	>6000
Star-4Split		50	96	144	435	-
Star-Butterfly		50	96	144	1047	-
Tiger		58	112	168	65	-
Kitten		64	122	184	48	-
Star-PNsplit	(Fig. 9b)	110	216	324	625	-
Snail-286	(Fig. 9c)	145	286	429	1315	-
Horse	(Fig. 9d)	152	302	452	946	-
Hand	(Fig. 9f)	170	336	504	1377	-
Dragon	(Fig. 9a)	172	344	514	1292	-
Bunny-348		176	348	522	976	-
Armadillo	(Fig. 9g)	195	386	579	730	-
Pooh		198	392	588	957	-
Moneybox-392	(Fig. 9h)	196	392	586	2200	-
Meister		200	394	592	3900	-

Table 1: Table showing the unfolding performance for different models.

Note that the performance of the present approach is not only influenced by the number of faces, but also by the size of Gluetabs. The bigger the Gluetabs are, the more iterations are needed to find a feasible unfolding. In the worse case, an unfolding might no longer be possible. Table 1 shows that not only the number of faces influences the computational time for finding a solution, but also shows that the randomness of simulated annealing lays an important role. For models marked with a "−" in Table 1, no unfolding were found within $100,000$ iterations. For a single iteration of the Tiger with 112 faces, the algorithm needs $1\mu s$ for the random move, $447\mu s$ to compute the MST, $149\mu s$ to unfold the mesh triangles, $1161\mu s$ to check for triangle-triangle overlaps, $287\mu s$ to select necessary Gluetabs, $33\mu s$ to unfold Gluetabs and $1180\mu s$ to check Gluetab-Gluetab and Gluetab-triangle overlaps resulting in overall $5018\mu s$. For the bunny model the respective times are $1\mu s$, $2638\mu s$, $752\mu s$, $10515\mu s$, $1984\mu s$, $124\mu s$ and $10016\mu s$ resulting in overall $39936\mu s$.

A comparison with a brute-force approach is conducted to investigate the feasibility of the solution space. Nonetheless, only small models can be solved using the brute-force approach. As the number of faces increases the computation using brute-force approach becomes infeasible.

6.2 Limitations and Discussion

According to our evaluation, multiple factors would limit the present approach. The present approach aims to solve the problem through considering the global optimum, while disregarding local overlaps. Thus, small local overlaps are harder to solve as their changes to $E(P)$ is rather insignificant.

Furthermore, meshes above 400 faces cannot be solved within $100,000$ iterations. Another limitation is that the Gluetabs that are necessary, cannot be calculated in advance. The number and the position of the Gluetabs depend on the unfolding.

7 Conclusion and Future Work

We present a new approach to unfold 3D Meshes as well as adding a minimal number of Gluetabs to aid users with reconstruction. Even with the addition of Gluetabs finding correct unfoldings is still possible using the suggested approach.

As for the future work, the quality of the unfolding can be improved by considering the spatial use of the unfolded patch to find compact space-efficient unfoldings.

Furthermore to help users with reconstruction the structures of the original mesh can be considered, such as the ears of a bunny are kept together in the unfolded patch. This should make the construction easier. To improve the performance and lower the impact of Gluetabs on the performance, the Gluetabs can be adjusted if the overlapping area is rather small. This would increase the solution space

for the problem. To compute the necessary modifications of the overlapping Gluetabs, the points of intersection can be used as the new boundary vertices of the Gluetab.

References

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

[2] 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.

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

[4] 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.

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

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

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

[8] 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.

[9] 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.

[10] Jonathan L Gross and Jay Yellen. *Handbook of graph theory*. 2004.

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

[12] Edwin Alexander Peraza Hernandez, Shiyu Hu, Han Wei Kung, Darren Hartl, and Ergun Akleman. Towards building smart self-folding structures. *Computers & Graphics*, 37(6):730–742, 2013.

[13] Scott Kirkpatrick, Daniel Gelatt, and Mario Vecchi. Optimization by simulated annealing. *Science*, 220(4598), 1983.

[14] 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.

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

[16] 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, 2004.

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

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

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

[20] Raphael Straub and Hartmut Prautzsch. Creating optimized cut-out sheets for paper models from meshes. *Karlsruhe Reports in Informatics 2011*, 36, 2011.

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

[22] 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.

[23] Shigeo Takahashi, Hsiang-Yun Wu, Seow Hui Saw, Chun-Cheng Lin, and Hsu-Chun Yen. Optimized topological surgery for unfolding 3D meshes. *Computer Graphics Forum*, 30:2077–2086, 2011.

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

[25] 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.