Simulated Annealing to Unfold 3D Meshes and Assign Glue Tabs

Thorsten Korpitsch
TU Wien
Vienna, Austria
e0152943@student.tuwien.ac.at

Shigeo Takahashi
University of Aizu
Fukushima, Japan
takahashi@acm.org

Eduard Gröller
TU Wien
VRVis Reasearch Center
Vienna, Austria
groeller@cg.tuwien.ac.at

Hsiang-Yun Wu
TU Wien
Vienna, Austria
wu@cg.tuwien.ac.at

ABSTRACT

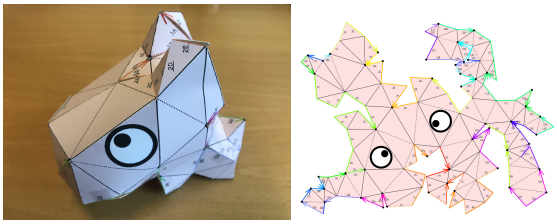
3D mesh unfolding transforms a 3D mesh model into one or multiple 2D planar patches. The technique is widely used to fabricate papercrafts, where 3D objects can be reconstructed from printed paper or paper-like materials. The applicability, visual quality, and stability of such papercraft productions is still challenging since it requires a reasonable formulation of these factors. In this paper, we unfold a 3D mesh into a single connected 2D patch. We also introduce glue tabs as additional indicators in order to provide users with extra space to apply glue for better reconstruction quality. To improve space efficiency, we do not apply glue tabs on every edge, while still guaranteeing the stability of the constructed paper model. A minimum spanning tree (MST) describes possible unfoldings, whereas simulated annealing optimisation is used to find an optimal unfolding. Our approach allows us to unfold 3D triangular meshes into single 2D patches without shape distortions, and employing only a small number of glue tabs. A visual indicator scheme is also incorporated as a post-process to guide users during the model reconstruction process. Finally, we qualitatively evaluate the applicability of the presented approach in comparison to the conventional technique and the achieved results.

Keywords

3D mesh unfolding, Simulated annealing, Glue tabs, Graph theory

1 INTRODUCTION

Papercraft is a popular art form, 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 is unfolded into a single or multiple 2D patches, which can then be printed and used to reconstruct the model in 3D. Possible models range from simple ones, such as paper aeroplanes, to complex ones, such as buildings. Recently, papercraft approaches are also used in combination with self-folding materials to form structures in an automatic fashion [1]. As demonstrated by Takahashi et al. [19], unfolding a 3D triangular mesh into a single patch, in contrast to multiple patches, eases the reconstruction process for users. However, finding 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 to be an NP-complete problem [8]. Reconstruction of a model can be hard even with indicators that show which edges should be glued together [19].



(a) (b)

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

Glue tabs are essential as they allow users to build clean 3D models and guide them during the reconstruction process. In this paper, we propose a technique to unfold a 3D mesh with a minimum number of glue tabs. This eases the reconstruction of models by guiding edges that should be glued together, as well as providing users sufficient surface to apply glue. The addition of glue tabs increases the complexity of finding an overlap-free unfolding, due to the combinatorial complexity of selecting edges where to attach glue tabs. We calculate minimum spanning trees of the dual graph of the 3D mesh, which describe possible unfoldings. Simulated annealing optimisation is used to find an overlap-free unfolding. Glue tabs are pre-calculated and treated analogue to mesh faces when unfolding the 3D model.

We incorporate trapezoidal glue tabs due to their popularity. Advantageously a trapezoid consists of two triangles, so that we can consider glue tabs as additional faces of the 3D mesh and apply similar overlap-detection when searching for a feasible solution. Furthermore, this shape gives users more space than a sim-

ple triangle. It is also preferable to a rectangle as it takes less space. As the glue tabs size increases, the difficulty of finding an overlap-free unfolding also increases. Our contributions include the following:

- Introduction of a minimum number of glue tabs to mesh unfolding problems.
- A new meta-heuristic approach to finding unfoldings of 3D meshes.
- Proof that the number of necessary glue tabs can be optimally pre-computed for each unfolding patch.

The remainder of this paper is structured as follows: Section 2 discusses conventional mesh unfolding and optimisation techniques. Section 3 describes the concepts used in this paper. In Section 4, we describe necessary steps of the method. Section 5 brings insight into the implementation. Section 6 shows several experimental results generated using our approach and evaluates quantitatively and qualitatively. 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 to the proposed approach.

2.1 Optimised Unfolding of 3D Meshes

Mesh unfolding has different applications, such as creating papercraft models [16, 19] and the creation of models from self-folding materials [6]. Some techniques employ mesh deformation [2, 12], in order to relax the problem. Mitani et al. [12] and Chang et al. [2] propose methods that allow mesh simplification as a pre-processing step. Also, the theoretical perspectives [14] of the problem have also been explored. Some authors study different types of target meshes, for example, orthogonal polyhedra [20]. The most relevant work to ours is by Takahashi et al. [19], who proposed a genetic-based algorithm to find a single connected patch for printing purposes. They unfold 3D models without distorting or editing the original 3D model. The key concept of unfoldability is borrowed from topological surgery, in order to guarantee an unfolded patch can be stitched together along the corresponding cut edges.

Contrary to the work done by Takahashi et al. [19] our paper explores a meta-heuristic simulated-annealing approach to find unfoldings. Simulated annealing is easy to implement compared to genetic algorithms and it is more likely to find an optimal solution compared to a greedy algorithm [16].

Another key conventional approach is investigated by Straub et al. [16]. They explored unfolding and attaching glue tabs on all cut edges. They also explored the removal of overlaps by introducing new subdivisions to the mesh. A greedy algorithm optimises the unfolding and resolves overlaps. Glue tabs that have been added to an unfolding are optimised, i.e. changed in size to avoid overlaps, after an initial unfolding has been found. For printability, the unfolding is then separated into multiple cut-out sheets if it does not fit on a single one.

In this paper, the proposed algorithm examines all possible glue tabs in advance and selects a minimum number thereof at each unfolding iteration. To the best of our knowledge, the simultaneous handling of unfolding and glue tabs has not been explored in the state-of-the-art. Our technique is an improvement of the approach studied by Takahashi et al. [19], where reconstructing a 3D model is facilitated if the mesh is unfolded into just a single patch.

2.2 Optimisation Techniques

Since the 3D mesh unfolding problem is an NP-complete problem [8], optimisation techniques are often used to find a solution. Trying all combinations would not be practical in most of the cases. Many optimisation techniques are well explored, including greedy algorithms [5] or heuristic optimisation techniques [11]. Simulated annealing is a well-known optimisation technique [9] that is widely applied in computer science [4] and other scientific fields [13, 18].

```
Data: Configuration P, Max Temperature t_max
Result: Optimised Configuration P
1 Set t = t_max;
2 while t > 0 do
      /* Random step to generate P' from P */
3     Create P' from P;
4     if energy(P') ≤ energy(P) then
5         Set P = P';
6     else if rand(0,1) ≥ exp(−(energy(P) − energy(P'))/k_B/t)
        then
7         Set P = P';
8     Decrease t;
9 end
```

Algorithm 1: The pseudo-code of simulated annealing [9].

Algorithm 1 depicts the concept of a simulated annealing approach, where a configuration P is optimised by minimising its energy. In each iteration, a new configuration P' is created, and the energy is compared to the previous configuration through the function $energy(*)$. In each iteration, there is a probability 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 experiments.

Optimising 3D Mesh unfolding with simulated annealing has the distinct advantage over greedy algorithms, that it is less likely to get stuck in a local minimum.

Compared to genetic algorithms, simulated annealing is easier to implement and is good at approximating a global optimum. Therefore, we chose simulated annealing as the optimisation approach in our unfolding.

3 DEFINITION OF KEY CONCEPTS

In this section, the terminology used in this paper is defined and key elements are described.

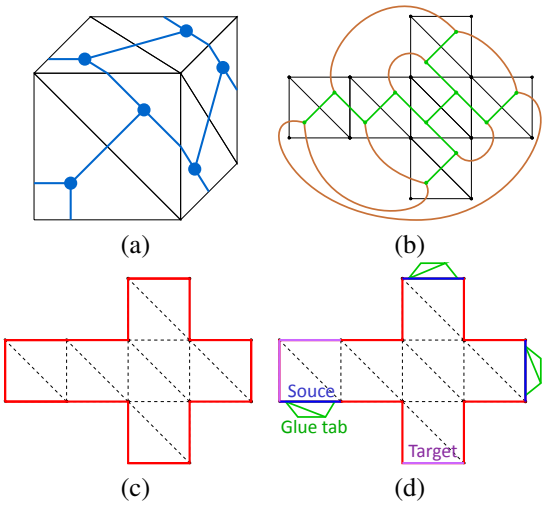


Figure 2: (a) A triangular mesh M (black) and its dual graph D (blue). (b) The MST T (green) and remaining edges (brown) in the dual graph. (c) Bend-Edges (dotted) and Cut-Edges (red). (d) Trapezoidal glue tabs (green) with source and target edges.

Triangular Mesh M. The input for the algorithm is a triangular mesh M (see black edges in Figure 2 (a)). A mesh M is defined as a triple (V_M, E_M, F_M), where V_M, E_M, and F_M represent the sets of vertices, edges, and faces, respectively.

Dual Graph D. Graph $D = (V_D, E_D)$ is the dual graph of the mesh M. For each face $f \in F_M$, a dual vertex in V_D is first assigned, followed by connecting dual vertices using a dual edge E_D, if the two faces that enclose the dual vertices are adjacent in M. In other words, the dual graph D has an edge if and only if the two corresponding faces of M are separated by an edge in the mesh [7].

The dual graph is then used to find an unfolding, since a dual edge connects neighbouring faces. A dual edge can either represent an edge that is cut or an edge that is used for folding the papercraft. Note that each triangular mesh has only one corresponding dual graph, which can be computed efficiently. This makes it a very compelling concept for 3D mesh unfolding.

Minimum Spanning Tree (MST) T. From the dual graph D, a MST T can be computed. Given a weight $c(e)$ to each edge $e \in E_D$, a minimum spanning tree $T = (V_T, E_T)$, where $V_T = V_D$ and $E_T \subseteq E_D$, is computed by minimising the cost function $\sum_{e \in E_T} c(e)$ [3]. The MST in combination with the dual graph is a viable concept to calculate possible unfoldings, as explained by Straub et al. [16]. To compute a MST, we need a weight for each edge in the dual graph. Initially, we assign a random weight between $(0, 1)$ for each edge in our experiment. In practice, the weights are adjusted to generate a better MST as the optimisation proceeds.

Unfolding. An unfolding is defined as the planar representation of a 3D mesh, and is computed by mapping faces onto a 2D surface. One can unfold the faces of a mesh one after another by referring to the MST T described previously. This process is called mesh unfolding. We follow the strategy of Takahashi et al. [19], and define that the goal of an unfolded mesh will not contain overlapping areas. A correct unfolding should have only occlusion-free areas (see Figure 2 (b)).

Bend-Edge. The dual edge of $e \in E_T$ is defined as a Bend-Edge (see dotted edges in Figure 2(c)), where users can fold their papercraft by bending Bend-Edges.

Cut-Edge. The dual edge of each $e \in E_D \backslash E_T$ (brown edges in Figure 2(b)) is called a Cut-Edge (red edges in Figure 2(c)). Glue tabs can be applied to Cut-Edges.

Glue Tab. A glue tab is defined as an additional region that does not exist in the original mesh. The idea of incorporating these regions has been widely used to apply glue for attaching Cut-Edges of a papercraft. It can have different forms, while in this paper, we introduce trapezoid glue tabs (Figure 2(d)) as our default setting, due to its popularity in the community.

4 UNFOLDING MESHES WITH ADDITIONAL GLUE TABS

Figure 3 shows the step by step (S1)-(S5) process of our approach. We will describe the high-level idea here, followed by the detail explanation in the corresponding subsections.

Once a triangular mesh M is loaded (Figure 3(a)), in (S1) the corresponding dual graph is computed (the blue graph in Figure 3(b)). In (S2), a MST T is computed (the green tree in Figure 3(c) and the Cut-Edges in red). Based on this T, a minimum number of glue tabs is selected (Figure 3(d)). The next step is to unfold the 3D mesh and assign glue tabs referring to the MST T (Figure 3(e)). If any overlaps occur, we create a different MST by changing an edge weight, and return and continue with (S2). Otherwise, we optimise the unfolding as described in Section 4.5, which results in an unfolded patch with better space utilisation (Figure 3(e)). Figure 3(f) shows additional visual indicators added during the post-processing. Steps (S1)-(S5) will be explained in the following subsections.

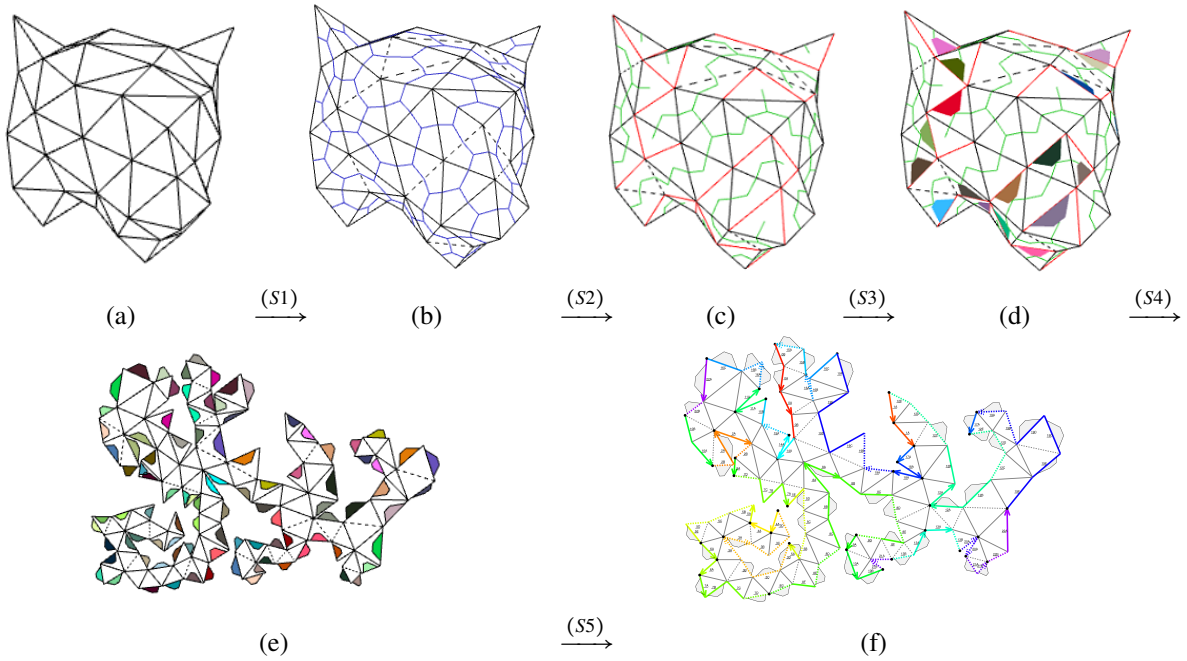


Figure 3: Steps (S1)-(S5) of the proposed unfolding process. (a) The input 3D mesh. (b) The dual graph (blue) of (a). (c) MST (green) and Cut-Edges (red) of (b). (d) Minimal number of glue tabs. (e) The optimised unfolding, and (f) the patch after the post-processing.

4.1 Dual Graph and Preparation (S1)

The dual graph D is computed initially (Figure 3(b)). We iterate through all faces of the mesh and save a dual edge for each pair of faces that share an edge. To compute a MST later, each new dual edge is initially assigned a random weight between $(0,1)$. Note that the weight can be also computed based on the angle bounded by the two adjacent faces or other measures. We choose a random approach here due to its better performance in comparison to other measures. More sophisticated measures, such as mesh structure analysis, will be conducted in a future study.

Glue Tab Pre-Computation

In this step, we also initialise glue tabs. After computing the dual graph, we compute all possible glue tabs, in order to link each dual edge to its corresponding glue tab. For each edge $e \in E_M$ a glue tab is computed. The height of the glue tab is 20% of the face size it will be glued to. The long base length is 80% and the short base length is 40% of the edge length it is attached to, as shown in Figure 2(d). These parameters are determined empirically to avoid an infeasible solution space.

4.2 MST Computation (S2)

A MST T is computed from the dual graph D using the well-known Kruskal's algorithm [10]. Recall that the dual edges of E_T are all Bend-Edges, while the dual edges $e \in E_D \backslash E_T$ are all Cut-Edges (red edges in Figure 3(c)).

4.3 Glue Tabs Selection (S3)

In our approach, we only select certain Cut-Edges to assign glue tabs. This is because assigning glue tabs on every Cut-Edge will limit the number of feasible solutions in the search space and increase the effort needed when reconstructing the 3D model. Therefore, we assign only a minimum number of glue tabs that will result in a stable constructed 3D model. For each vertex, we allow at most one edge associated with this vertex without a glue tab, so that the created papercraft will not contain an open hole after being built. This is achieved by visiting all pre-computed glue tabs and selecting a minimum set of them based on the lemma below.

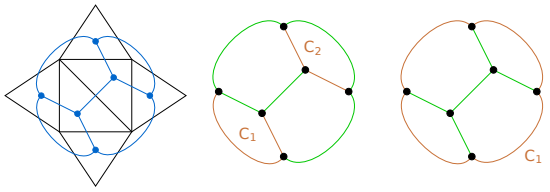
This implies that each vertex $v \in V_M$ of a mesh can only be incident to one edge $e \in E_M$ that is a Cut-Edge, and will not be fixed with a glue tab. This stability is reasonable because adding more glue tabs will not yield a more stable reconstructed model, while increasing the reconstruction effort as shown in our user study.

Minimum Number of Glue Tabs

The aforementioned concept can be summarised as:

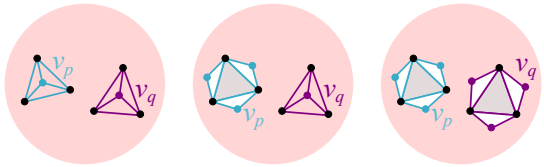
Definition 4.1. Given a mesh $M = (V_M, E_M)$ and the corresponding dual graph $D = (V_D, E_D)$, the graph $C = (V_C, E_C)$, where $V_C = V_D$ and $E_C = E_D \backslash E_T$, contains only Cut-Edges that are not assigned a glue tab. If $\forall v \in V_C \mid degree(v) \leq 1$ holds true, the reconstruction of the 3D model is stable. The function $degree(v)$ returns the vertex degree of a vertex v.

Figure 4 gives an example to demonstrate this property. The input mesh M (black) and the corresponding dual graph D (blue) are shown in Figure 4(a). Different MST T (green trees in Figure 4(b) and (c)) can be synthesised from D. The graph C in Definition 4.1 can be determined as follows. Assume that the graph C is the union of n connected components $C_i = (V_{C_i}, E_{C_i})$, where $i = 1,..,n$ and $1 \leq n < |E_C|$. For each connected component C_i we assign glue tabs, under the condition that $\forall v \in V_C \mid degree(v) \leq 1$ holds.



(a) (b) (c)

Figure 4: (a) An unfolded mesh M (black) and its dual graph D (blue). (b) A MST (green) of (a) and its E_C resulting in two path components C_1 and C_2 (brown). (c) Another MST T (green) of (a) that defines the unfolding and its corresponding E_C as a cycle component C_1 (brown).



(a) Closed mesh (b) One hole (c) Two holes

Figure 5: An example of multiple cycles.

We first investigate the properties of C_i. Suppose we have a closed manifold mesh. Then there are two types of C_i that can be taken into consideration. One type is a path (brown graph in Figure 4(b)), and the other one is a cycle (brown paths in Figure 4(c)). This is because the dual vertex of a face in a triangular mesh connects to three dual edges. At least one dual edge belongs to the MST, and thus each vertex in C_i can have a maximum degree of two. We further summarise the combinations of cycles and paths into three cases. Case 1 contains only one cycle, Case 2 contains only paths, and Case 3 contains cycles and paths. Figure 5 shows the reason why the cycles-only case does not exist. A cycle C_i happens only when we cut a mesh into multiple patches, or open a vertex. If we open vertices on a closed manifold mesh, the Euler Formula does not hold: $(V_{C_i} - n) + \sum_{p=1}^{n} degree(v_p) - (E_{C_i} + \sum_{p=1}^{n} degree(v_p)) + F_{C_i} \neq 1$. This leads to the condition that our unfolded patch is not a single connected planar graph, which violates our assumption.

Since C_i can be only a cycle or a path, the glue tab assignment can be done independently. In practice, the problem is equal to retrieving a minimum edge cover of a cycle or a path, which can be calculated in $O(|E_{C_i}|)$ time.

Based on this fact, we can devise an algorithm that finds the minimum number of glue tabs for C_i. In Case 1, C_i is a path, as shown in Figure 4(b). We pick a vertex $v \in V_D$ with $degree(v) = 1$ as the start vertex, then we iterate over the path and attach a glue tab on every second edge. More specifically, the number of glue tabs necessary of a path is exactly $f(C_i) = \lfloor \frac{|E_{C_i}|}{2} \rfloor$. In Case 2, if C_i is a cycle, as shown in Figure 4(c), we can pick an arbitrary starting edge for the path and the number of glue tabs can be computed as $f(C_i) + 1$.

Lemma 4.1. *The minimum number of glue tabs for a mesh M can be computed in $O(n)$ time. The total glue tab number is $\sum_{i=1}^{n} f(C_i)$, where n is the number of connected components defined in Definition 4.1.*

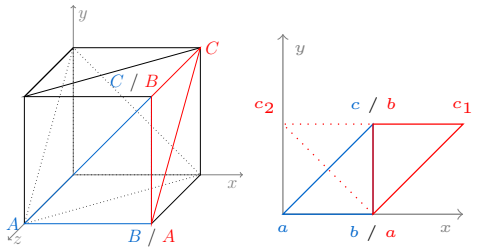
Proof. Note that $f(C_i)$ returns the minimum number of glue tabs, can be proven by induction. Let $P(n)$ be the statement that $f(C_i) = \lfloor \frac{|E_{C_i}|}{2} \rfloor$ determines the minimum number of glue tabs necessary for a path. The base case is $P(1)$. For $|E_{C_i}| = 0$, $\lfloor \frac{0}{2} \rfloor = 0$ is true. Let us consider the step from n to $n+1$ and assume that $P(n)$ is true. Suppose $n+1$ is odd, this means that n is even and $P(n+1) = P(n) + 1$, in case n is odd, $P(n+1) = P(n)$. This leads to the fact that $P(n+1) = \lfloor \frac{|E_{C_i}|}{2} \rfloor$ holds true. The case of a cycle can be proven similarly. Based on this fact, we now show that $\sum_{i=1} f(C_i)$ is minimal by contradiction. We assume that $\sum_{i=1} f(C_i)$ is not minimal, which implies that $\exists f(C_i)$ having a larger value than its minimum. This constitutes a contradiction to our previous statement. ■

4.4 Mesh unfolding using the MST (S4)

Figure 6 shows how an unfolding of the mesh is computed. Each face is defined by three points A, B, and C in 3D and a, b and c in the 2D representation. For the first face, as shown in blue in Figure 6, we set the vertex a to $(0,0)$, b to $(0,\overline{AB})$. Then we calculate the position of c using the following Equation 1.

$$\begin{aligned} s &= \|(B-A)\times(C-A)\|/(\overline{AB})^2 \\ d &= (B-A)\cdot(C-A)/(\overline{AB})^2 \\ c_x &= a_x + d(b_x - a_x) - s(b_y - a_y) \\ c_y &= a_y + d(b_y - a_y) + s(b_x - a_x) \end{aligned} \tag{1}$$

For every following triangle we already have the positions of two vertices and only need to calculate the position for the last vertex c, which has two possible positions c_1 and c_2 as shown in Figure 6(b). We select the position that is on the opposite side of the shared edge. If a glue tab is attached to a face it is unfolded analogue to mesh triangles.



(a) (b)

Figure 6: (a) A 3D model of a box, and (b) the unfolding of the first two faces.

4.4.1 *Overlap Detection*

After an unfolding is found, we detect overlaps to determine if the unfolding is correct. An overlap is defined if two faces are intersected partly with each other, or if a face is entirely located inside another face. For glue tabs, the definition is similar. In total, three different cases of overlaps can occur, including either face-face, glue tab-glue tab, or face-glue tab overlaps.

If an overlap occurs, the Sutherland-Hodgman Clipping algorithm [17] 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 p. The area is calculated with the shoelace formula $Area = \frac{1}{2}\left|\sum_{i=1}^{k-1} x_i y_{i+1} + x_k y_1 - \sum_{i=1}^{k-1} x_{i+1} y_i - x_1 y_k\right|$ [15], where x_i and y_i are the coordinates of the i-th point in p. This area is then used as the $energy(P)$ function to describe the quality of the unfolding.

Face-face overlaps are checked before glue tab related overlaps are checked to improve the performance. This is because if faces overlap with each other, an overlap-free unfolding cannot be found even if the glue tabs are overlap-free. Moreover, the performance can be improved by limiting the number of faces that need to be checked. The overlap detection does not need to be done if faces share Bend-Edges, as no overlap is possible due to how we handle the unfolding.

4.4.2 *Spatial Optimisation*

After the simulated annealing process comes to an end, we try to find an unfolding that enables better spatial efficiency, see Figure 7. The energy-function is thereby redefined as $energy(P) = spread(\text{x-axis}) + spread(\text{y-axis})$, where $spread(\text{x-axis})$ measures the distance of the vertex with the highest value on the x-axis minus the vertex with the lowest value on the x-axis. We use simulated annealing again, whereas the algorithm now discards all unfoldings with overlaps.

4.5 Post-Processing (S5)

Multiple strategies are introduced to improve the visual quality, namely the clean look of the reconstructed model and guide users with reconstruction using visual indicators.

Colour Coding and Edge Numbering. According to the stitching algorithm proposed by Takahashi et al. [19], boundary edges that will be glued together, are assigned the same colour. We follow their strategy to generate our visual indicators.

Bend-Edge Coding. Each Bend-Edge is coded to distinguish between a mountain-fold or a valley-fold, as proposed by Takahashi et al. [19]. A mountain-fold is represented by a solid line and a valley-fold is displayed by a dotted line (see Figure 3(c)).

5 IMPLEMENTATION DETAILS

The system is implemented using an Intel Core i7 CPU (4x3.3 GHz, 4MB L3 Cache) and 8 GB RAM. The source code is written in C++17 using OpenGL ver. 4.5 to visualise the results and CGAL ver. 4.13 to manage the graph data structure. QT 5.13 is employed to provide a graphical user interface, and CMake ver. 3.14 is used to build the process. GCC 9.2 is used to compile the sources on Ubuntu 18.04.02 LTS.

We limit in the simulated annealing process the maximum number of iterations to $t = 100{,}000$ and set $k_B = 0.002$ based on our empirical experience.

Then we employ the simulated annealing process. Additionally, a face-face overlap is weighed ten times higher than a glue tab-glue tab and a glue tab-face overlap. This prioritisation should force the algorithm to solve the mesh unfolding first, as glue tab related overlaps could be resolved using post-processing.

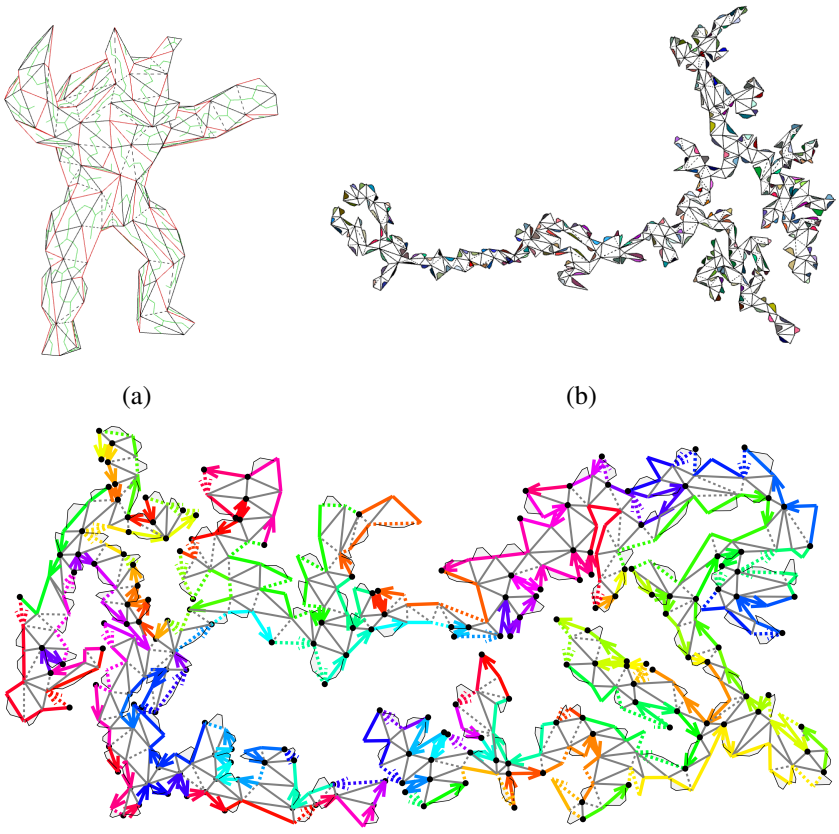
At the end of every iteration, if P is set to P', we visualise the unfolding. The annealing process terminates if $energy(P') = 0$. If until $t = 0$ we do not find a correct unfolding we terminate unsuccessfully. The spatial optimisation is limited to $5{,}000$ iterations.

6 RESULTS AND EVALUATION

In this section, we show several experimental results and discuss the findings in our evaluation.

6.1 Experimental Results

Figures 3, 7 and 8 show the results generated using our system. In Figure 7, we compare the unfolding results that are considered without and with spatial optimisation. In Figure 7(b), the triangles are smaller than the ones in (c). Figure 8 show other interesting models, where the glue tabs are highlighted in the 3D meshes. The testing data specification is summarised in Table 1.



(a) (b)

(c)

Figure 7: (a) Armadillo mesh. (b) Unfolding. (c) Spatially optimised unfolding.

6.2 Quantitative Evaluation

Figure 8(a)-(c) shows different models and their unfoldings. Sufficient results can be achieved within 100,000 iterations limit to find an unfolding. With a limit of 5,000 iterations for spatial optimisation compact results can be achieved. More iterations did not yield consistently more compact results.

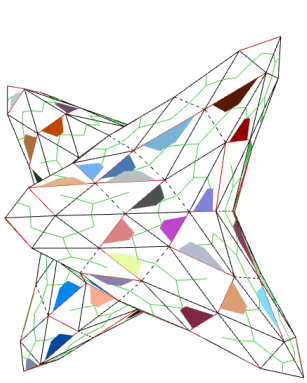
Glue tabs certainly impact the amount of time necessary to find an unfolding, but they are not the only influence. The structure of the model also impacts the time necessary to find an unfolding. While some models have a similar number of faces, the time to unfold can differ substantially. This can be attributed to the geometric properties of the models, as spherical models seem to unfold consistently faster than others. Further, the time discrepancy can be attributed to the random-walk of the simulated annealing process, where the time to unfold can greatly differ for the same model.

To evaluate the algorithm, we conduct an experiment by unfolding a variety of 3D models. The results are summarised in Table 1 using glue tabs as described in Section 4.1. In the experiment, a maximum of 100,000 iterations is used to limit the computational time, as an experimental value.

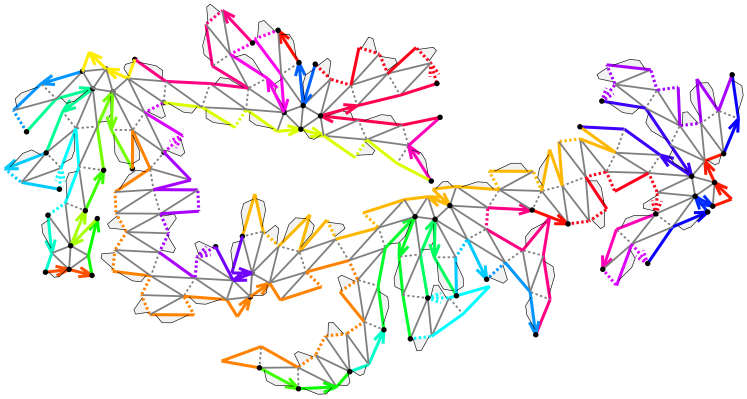
Table 1: Table showing the average unfolding performance for different models. V_M is the number of vertices, E_M the number of edges and F_M the number of faces.

Model		V_M	E_M	F_M	Time (seconds) Ours	Brute-force
Icosa		12	30	20	0	0
Star		14	36	24	8	19
Star-Sqrt3		38	108	72	31	>6000
Tiger	(Fig. 3)	58	168	112	65	-
Star-PNsplit	(Fig. 8(a))	110	324	216	625	-
Horse	(Fig. 8(b))	153	453	302	946	-
Hand	(Fig. 8(c))	170	504	336	1377	-
Bunny-348		176	522	348	976	-
Armadillo	(Fig. 7)	195	579	386	730	-
Moneybox-392		196	586	392	2200	-

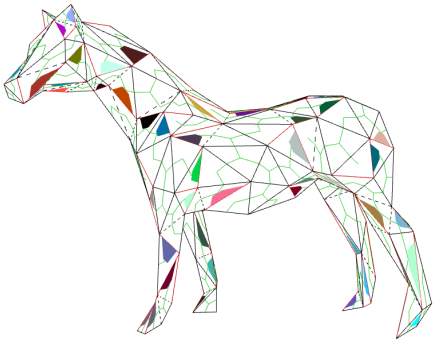
The performance of the presented approach is not only influenced by the number of faces, but also by the size of the glue tabs. The bigger the glue tabs are, the more iterations are needed to find a feasible unfolding. In the worst case, an unfolding might no longer be possible. Table 1 shows that not only the number of faces influences the computational time for finding a solution, but also shows that the randomness of simulated annealing plays an important role. A comparison with a brute-force approach is conducted to investigate the feasibility of the solution space. The brute-force approach cal-



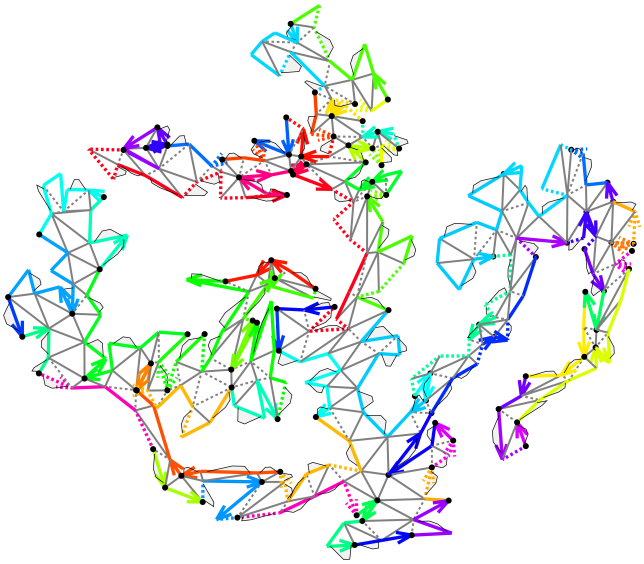
(a.1)



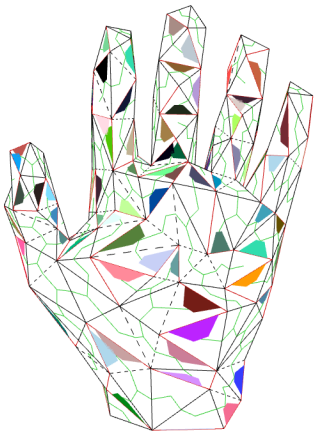
(a.2)



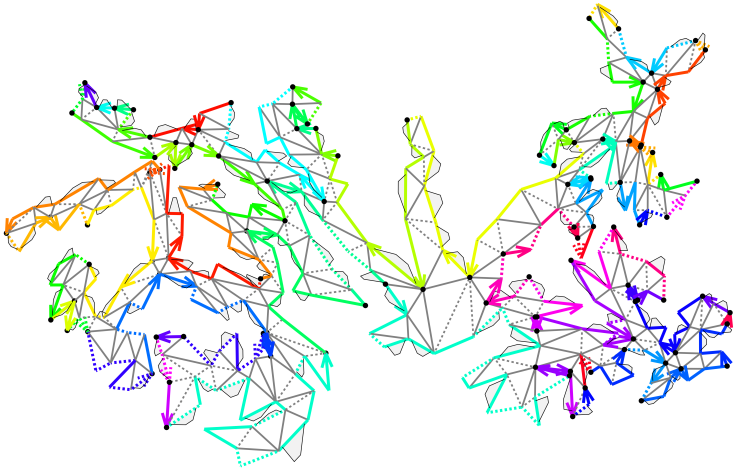
(b.1)



(b.2)



(c.1)



(c.2)

Figure 8: (a.1) Star mesh with 216 faces. (a.2) Unfolding of the star mesh. (b.1) Horse mesh with 302 faces. (b.2) Unfolding of the horse mesh. (c.1) Hand mesh with 336 faces. (c.2) Unfolding of the hand mesh.

culates all possible minimum spanning trees, as well as all possible glue tab positions until a solution is found. Only small models can be solved using the brute-force approach. As the number of faces increases using the brute-force approach becomes infeasible.

6.3 Qualitative Evaluation

To evaluate the impact of glue tabs on the reconstruction process as well as the visual quality of the reconstructed models, we conducted a user study with five participants (P1)-(P5) between the age 22 and 32 with different educational backgrounds. Three of them have a computer science background, and one has a finance background. The participants were asked to construct the tiger model (Figure 9) twice. Once is with glue tabs (Figure 3(f)), and the other time without. Apart from the glue tabs, both unfoldings have the same visual indicators, as explained in Section 4.5. Models with glue tabs were prepared with double-sided tape, where participants still needed to remove the protective covering. For models without glue tabs, we prepared glue strips. At the beginning of the study, participants were first instructed on how to construct the papercraft. We prepared an unfolded octagon with glue tabs and visual indicators, which the participants used to practice how to follow the visual indicators. To avoid a learning effect, P2-P4 started with the model with glue tabs (O1), and P1 and P5 began with the model without glue tabs (O2).

Table 2: Time (in minutes:seconds) for participants to reconstruct models. M/F marks the gender of the participant. Bold values are the time achieved in the first round. † indicates which model participants found easier to reconstruct, while ‡ marks the preferred visual model.

	P1 (M)	P2 (M)	P3 (F)	P4 (M)	P5 (M)
w. glue tabs	26:15	**23:10**‡	**14:08**†‡	**19:08**‡	22:15†‡
wo. glue tabs	**44:40**†‡	18:47†	19:09	15:04†	**failed**

Table 2 shows the time spent for each model. Most participants required less time for the second model reconstruction, while P3 spent more time for the second one. Her second model does not have glue tabs, which implies that the model with glue tabs is sufficiently easier for her. P5 failed to construct the model without glue tabs after trying for an hour. We further analyse and compare the time reduction of the two model orders, which suggests that glue tabs improve the amount of time necessary for reconstruction. The time needed for reconstruction for P1 decreased by 18:25 and for P4 by 4:04, who started without and continued with glue tabs respectively, while it increased for P3 by 5:01 and for P2 by 4:23, who both started with glue tabs.

We also asked participants, which model was easier to create. Opinions of participants differed. P1, P2, and P4 said that the model without glue tabs was easier, although P1 spent almost half time for the glue tab model. P2 remarked that without pre-cut glue stripes they would find the reconstruction with glue tabs easier. P3 emphasised that the glue tab model is significantly easier, as fewer edges need to be glued and edges are easier to connect. However, the last glue tab is problematic, since it cannot be fixed from inside.

We also asked participants to evaluate which constructed model has a better visual appearance. P2, P3, P4 and P5 perceived that the model with glue tabs is cleaner, while P1 prefers the model without glue tabs. P1 stated that it is easier to use stripes in the process.

Figure 9 shows the results from P3. We highlight two parts of the tiger. With glue tabs, sometimes the edges do not stick together well, as shown at the upper part of the picture, while edges fixed with glue strips are less problematic. On the contrary, in the lower part, the glue tabs stick together well.

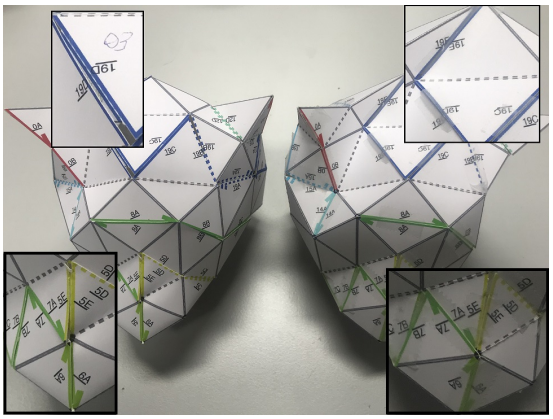


Figure 9: Tiger with glue tabs (left) and without (right) reconstructed by P3.

6.4 Limitations and Discussion

Multiple factors could limit the presented approach. The approach tends to solve the problem in global way, while disregarding local overlaps. Thus, small local overlaps are harder to solve as their changes to $energy(P)$ is rather insignificant. Another limitation is that the current approach is not globally optimal, since we initially limit the glue tab placement to a certain degree. One limitation inherited from the glue tab design is that one can hardly connect the last two faces, since users cannot access the inside of the model anymore to apply counter-pressure.

7 CONCLUSION AND FUTURE WORK

In this paper, we present a new approach to unfold 3D meshes, by adding a minimal number of glue tabs to aid users with the reconstruction. We also demonstrate the applicability of the approach.

To improve the performance and lower the impact of glue tabs on the performance, a glue tab can be adjusted

if an overlapping area is rather small. This would increase the solution space for the problem.

In the future, we plan to incorporate mesh structure analysis to guarantee that semantic structures, such as the ears of a bunny, are kept together in the unfolded patch. This could improve the reconstruction process.

ACKNOWLEDGEMENTS

This paper was partly written incollaboration with the VRVis Competence Center. VRVis is funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency in thescope of COMET - Competence Centers for Excellent Technologies (854174), which is managed by FFG.

REFERENCES

[1] Byoungkwon An, Ye Tao, Jianzhe Gu, Tingyu Cheng, Xiang 'Anthony' Chen, Xiaoxiao Zhang, Wei Zhao, Youngwook Do, Shigeo Takahashi, Hsiang-Yun Wu, et al. Thermorph: Democratizing 4d printing of self-folding materials and interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

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

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

[6] 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.

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

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

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

[10] 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.

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

[12] 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.

[13] 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.

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

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

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

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

[18] 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.

[19] 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.

[20] 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.