Learning to Segment and Unfold Polyhedral Mesh from Failures

Zhonghua Xi[a], Yun-hyeong Kim[b], Young J. Kim[b], Jyh-Ming Lien[a,b]

[a]*Department of Computer Science, George Mason University, USA*
[b]*Department of Computer Science and Engineering, Ewha Womans University, Republic of Korea*

Abstract

Folding planar sheets to make 3D shapes from is an ancient practice with many new applications, ranging from personal fabrication of customized items to design of surgical instruments for minimally invasive surgery in self-folding machines. Given a polyhedral mesh, unfolding is an operation of cutting and flattening the mesh. The flattened polyhedral *nets* are then cut out of planar materials and folded back to 3D. Unfolding a polyhedral mesh into planar nets usually require segmentation. Either used as a preprocessing step to simplify the mesh and provide semantics or as the result of unfolding to avoid overlapping, the segmentation and the unfolding operations are decoupled. Consequently, segmented components may not be unfoldable and unfolded nets usually provide no semantic meaning and make folding difficult. In this paper, we propose a strategy that tightly couples unfolding and segmentation. We show that the proposed method produces unfoldable segmentation that resembles carefully designed paper craft. The key idea that enables this capability is an algorithm that learns from failed unfoldings.

Keywords: unfolding, folding, segmentation, shape analysis, paper craft, mesh processing

1. Introduction

Making 3D shape from planar sheets is an ancient practice with many new applications, ranging from personal fabrication of customized items [1], which is fueled by the recent maker movement, to design of specialized instruments in self-folding machines [2] mostly due to the advances in active materials. One of the prevailing methods for creating 3D objects from planar materials is "unfolding and folding" [3].

Unfolding involves cutting a given polyhedral mesh into surface patches and then flattening them. To ensure that a surface patch can be flattened, existing methods [4, 5] either approximate the patch by developable surfaces or ensure that the patch forms a *net*, i.e., a patch that can be cut and flattened by rotating its facets along one of the incident edges without overlapping with other facets [6]. The flattened patches are then cut out of planar materials and folded back to 3D.

Either cutting a polyhedral mesh into nets or approximating with developable surfaces, segmentation of the mesh (a process of breaking a mesh into multiple components) is usually involved; either before the unfolding algorithm is applied or as a product of the unfolding algorithms. Segmentation before unfolding is used as a preprocessing step to provide simplicity, as well as semantics [7, 5]. As shown in Fig. 1, segmentation is also a common technique used by paper craft designers. In the literature, shape segmentation is usually done without considering foldability [8, 9]. Consequently, surface patches produced by shape segmentation may still be cut into multiple nets or approximated by multiple developable surfaces which lose the semantic meaning and make folding and assembly less intuitive thus time-consuming.

Segmentation can also be produced in order to avoid overlapping in the nets [10]. However, these nets often provide little shape information. Examples of these nets produced by the existing methods can be found in Figs. 9, and 10. In both scenarios, segmentation and unfolding operations has been decoupled.

Figure 1: A commercial paper craft designed by KitRex [11] that shows segmented parts with anatomic meanings.

In this paper, we propose a strategy that produces polyhedral nets by tightly coupling the edge unfolding and surface segmentation operations. Our objective is to algorithmically produce nets that resemble carefully designed paper craft such as those shown in Fig. 1. Fig. 2 shows an example output of the proposed method. We show that the proposed method naturally provides semantic segmentation of the input mesh by unfolding the entire mesh multiple times. Even though most likely, all of these unfoldings will contain overlaps, the proposed method learns from these failures and identifies parts that may be unfolded into valid nets.

Existing shape segmentation methods rely heavily on shape features, such as curvature and geodesic distance. On the contrary, the proposed method creates the segmentation directly from information obtained from edge unfolding, therefore, ensures that

Figure 2: **Top row**: The "Dancing Children" statue (3000 triangles) is segmented into 16 parts by the proposed method. Most of these parts corresponds to the heads, torso, legs of the model. The unfolded nets are shown in the middle. **Second row**: Nets and the folded parts. **Third row**: A crafted paper model, which is about 30 cm wide, 11 cm deep and 22 cm tall. **Bottom row**: an optimization method proposed by Takahashi et al. [12] segments the same model into 21 nets that do not provide semantic information.

every component in the segmentation can be unfolded into *a single net* and maintain its semantics. An overview of the proposed method can be found in Fig. 6 and we will discuss the details in Section 4. Our experimental results demonstrate that our method provides advantages of both foldability and semantics, thus enables users to fold more complex models in shorter time. As shown in Section 5, we are able to fabricate models that are topologically more complex and contain much more facets than those reported in the literature.

2. Background

2.1. Polyhedra Unfolding

Mathematicians spend centuries in answering a question: given a polyhedron, is it always possible to unfold the polyhedron by cutting on the surface of it, such that the unfolding of the polyhedron does not contain overlapping? When cuts are restricted only to the edges of polyhedra, it becomes an edge-unfolding problem. For edge-unfolding, counterexamples were found for non-convex polyhedra; for convex polyhedra, though promising, it still remains open [13]. Later, heuristic methods were proposed in the literature for unfolding convex polyhedra to nets [6]. However, it becomes much harder to generate a single net for non-convex shapes. Both Straub and Prautzsch [10], Takahashi et al. [12] generate more than one connected components for complex non-convex shapes to avoid overlapping. The former one splits the unfolding when overlaps were detected while the later one first splits the mesh into multiple pieces then tries to merge them into one piece. All aforementioned works generate nets as final results, however, whether there exists a continuous folding motion that transforms the net back to its original shape is not considered in their works.

2.2. Paper Crafting via Shape Segmentation

Unfolding a non-convex polyhedron into a single connected component is hard but not necessary for certain applications, such as paper crafting: making 3D models from flat sheets of paper. For paper crafting, shape segmentation techniques were employed to simultaneously decompose and approximate the mesh into smaller pieces [4, 7, 5] such that the unfolding problem becomes solvable and the approximated 3D model can be obtained by assembling the folded shapes together. These approaches decompose the mesh into a few patches and approximate each patch with a strip, a generalized cylinder or a developable surface. Common drawbacks of these methods are that they could generate an arbitrary number of pieces and the cuts can be at arbitrary locations on the mesh which make assembling much harder and less fun, also the approximation ability is limited. Another category of shape decomposition method worth noting is called Nearly Convex Decomposition (NCD) [14, 15, 16], which segments a mesh into a controllable number (usually small) of part-aware components that are nearly convex. Mesh convexification shows great advantages in unfolding and continuous folding, and we can obtain either exactly the same model as the original one by assembling folded nearly convex patches, or an approximated model (with bounded error) by folding the nets generated from the convex hulls of those patches.

3. Preliminaries

3.1. Net of Polyhedra

An unfolding of a 3D polyhedron by cutting the surface of the polyhedron is called the *net* of the polyhedron if the flattened faces do not overlap in 2D. In this paper, we are interested in edge unfolding in which cuts are only allowed along the edges of the surface. An edge unfolding can be obtained by finding a spanning tree of the dual graph of the mesh (see proof in [6]). For a polyhedron with $|F|$ faces, there will be $|F| - 1$ edges in the spanning tree, called *fold edges*. All the rest of edges are

cut edges, such as the red edges in the heart shaped meshes in Fig. 4. Cut edges must first be cut in order to unfold the polyhedron and then later glued back to form the 3D shape. Heuristic methods have been developed to assign weight to each dual edge accordingly such that the minimum spanning tree of the dual graph has a high probability of becoming a net. Fig. 4 shows nets obtained by two heuristic methods and their corresponding spanning trees of the mesh's dual graph.

3.2. Heuristic Methods for Finding Nets

Schlickenrieder [6] comprehensively studied different heuristics for unfolding convex polyhedra to nets. Here we briefly introduce two most powerful heuristic methods that will be used in this paper for generating unfoldings.

Steepest Edge Unfolding The idea of this heuristic is to find steepest cut paths from v_- to v_+ w.r.t a given unit vector $\vec{c} \in \mathbb{R}^3$, where v_- and v_+ are bottom vertex and top vertex w.r.t to $\vec{c}$. For each vertex (except v_+), find the steepest edge $\vec{e}$ w.r.t to $\vec{c}$ and add e to the cut set $\mathbb{S}$. For all edges in $\mathbb{S}$, assign weight 1 to their corresponding dual edges, and assign weight 0 to rest of the dual edges. Unfoldings generated by this heuristic usually has a flower like shape. See Fig. 3 and the top example in Fig. 4.

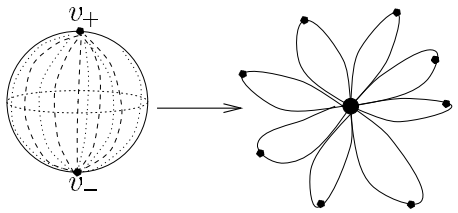


Figure 3: Steepest edge unfolding. $\vec{c} = \overrightarrow{v_- v_+}$

Due to its 100% success rate in unfolding convex polyhedra used in [6], Schlickenrieder [6] conjectured that Steepest edge unfolding is able to unfold all convex polyhedra. Though carefully constructed counterexamples were discovered [17], it remains a practical unfolder for convex meshes.

Flat Tree Unfolding Similar to *Steepest Edge Unfolding*, this heuristic assigns edge weight according to its *steepness* w.r.t a given unit vector $\vec{c} \in \mathbb{R}^3$, $weight(\vec{e}) = \frac{\langle \vec{e}, \vec{c} \rangle}{\|\vec{e}\|\|\vec{c}\|}$, where $\langle ., . \rangle$ indicates a dot production operation.

Though these two heuristics share the same idea, but the results yield by them are quite different as shown in Fig. 4. In our experiments, we found that *Steepest Edge* works perfectly on convex shapes but poorly on non-convex ones. *Flat Tree* has a lower success rate in finding nets for convex shapes, but it performs much better on non-convex ones than the steepest edge method. Note that, these heuristic methods are designed for convex polyhedra, though it might still work on some non-convex polyhedra, it can easily fail on very simple non-convex polyhedra like the star shape shown in Fig. 5.

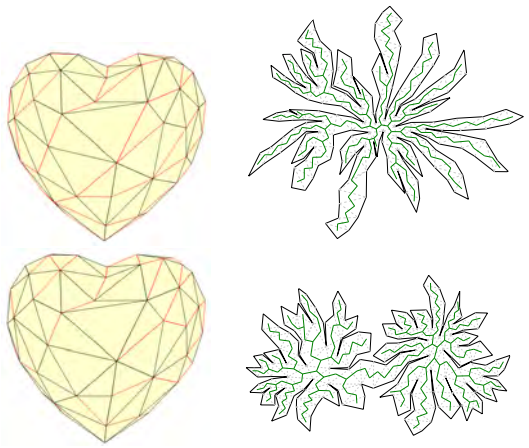


Figure 4: Two nets of a heart model (198 triangles) produced by the steepest edge heuristic (upper right) and flat tree heuristic (bottom right). These nets are produced by the minimum spanning trees (shown in green) of the edges weights assigned by the heuristics. Cut edges are shown in red.

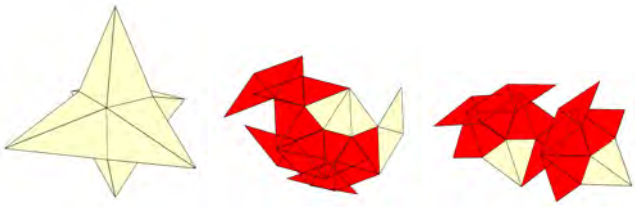


Figure 5: Unfoldings generated by heuristic algorithms for a non-convex polyhedron with overlaps. From left to right: star shape model; Steepest Edge Unfolding; Flat Tree Unfolding. Overlapped faces are shown in red.

4. Simultaneous Unfolding and Segmenting

Segmentation and unfolding are both edge-cutting operations that determine the foldability of a mesh, thus should not be decoupled. In this section, we will discuss a method that first estimates the likelihood of every pair of faces that can be unfolded together without overlapping, and then segments the mesh into face clusters that have high probabilities of becoming valid nets. Upon the failure of unfolding a cluster in the segmentation, the cluster is further segmented using the learned likelihood until all clusters are unfolded. Fig. 6 provides an overview of the proposed method.

4.1. Learn from Failed Unfoldings

To determine the likelihood of every pair of faces that can be unfolded together without overlapping, the proposed method unfolds the mesh multiple times using existing heuristics, such as steepest edge for example. After the mesh is unfolded m times using these unfolding heuristics, the proposed method analyzes whether two faces f_1 and f_2 belong to the same connected component without overlapping in all of these unfoldings. If f_1 and f_2 belong to the same non-overlapping connected component n times among m trials, then their foldability likelihood is simply n/m. Thus, the result of this learning step is a symmetric similarity matrix, called "foldability matrix", in which a large

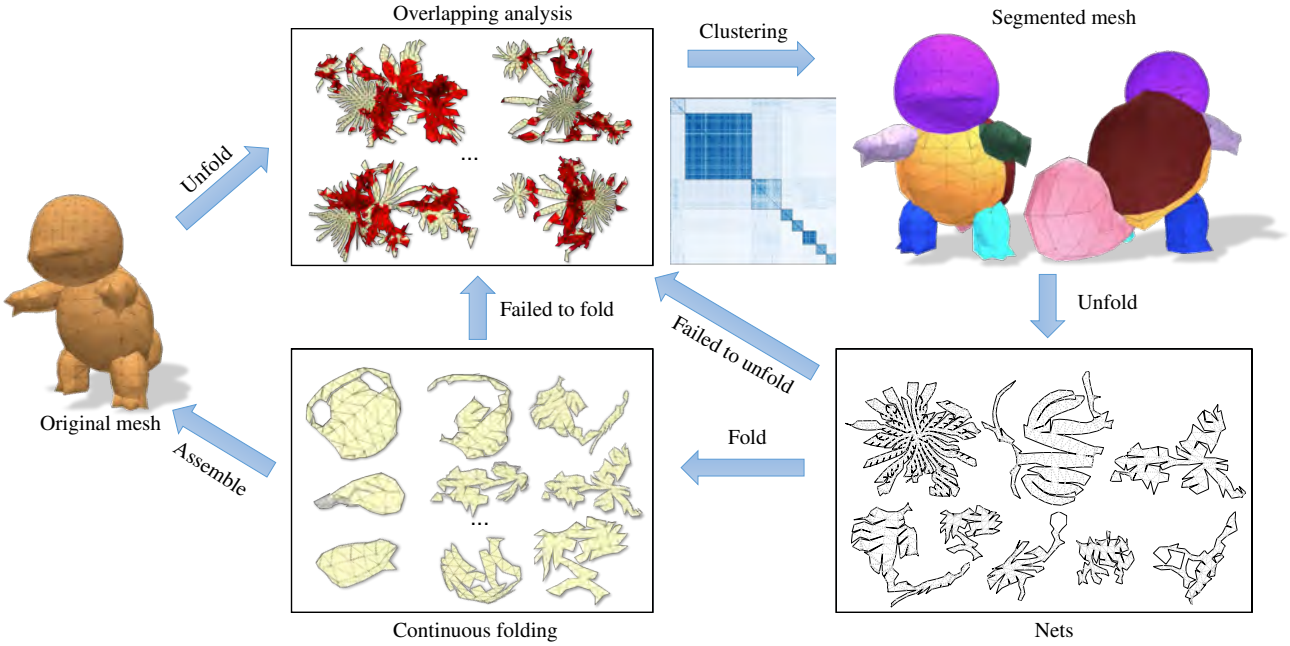


Figure 6: Overview of the proposed method. Overlapped faces are shown in red in the 'Overlapping analysis' box. Foldability matrix after clustering is shown below the 'Clustering' arrow, in which rows are sorted by cluster id. Pixel $p(i, j)$ indicates the probability that face f_i does not overlap with face f_j in the unfoldings, the darker the higher. Each block along the diagonal represents one cluster. If any of segment was failed to unfold to net or failed to fold back, we can further segment it using the proposed method. This process is repeated until all segments have nets and can be continuously folded back to 3D.

value means that two faces are likely to be unfolded without overlapping. We next describe each of these steps in detail.

4.1.1. Unfold the Mesh Multiple Times

As discussed earlier, an unfolding of a mesh is closely related to the minimum spanning tree of the dual graph of the mesh. That is, a set of edge weights determines a specific unfolding. Therefore, m unfoldings can be obtained by m sets of edge weights. If the random unfolding heuristic is used, then m unfoldings can simply be created by drawing $m||E||$ arbitrary numbers. If steepest edge or flat tree heuristics are used, then we draw m random unit vectors and use these vectors to determine the edge weights. Alternatively, instead of drawing random vectors, we also experimented with the surface vectors of the mesh, such as outward normal vectors of faces and vertices and vectors parallel to the edges. We found no differences of how the vectors are selected. The number of vectors plays a more influential factor.

4.1.2. Analyze an Unfolding

An edge unfolding usually contains multiple overlaps and is not a valid net. However, we can still obtain valuable information about the foldability of the mesh from an invalid net.

Given an edge unfolding represented by a tree $\mathcal{T}$, and let $\mathcal{L} = \{(f_i, f_{j\neq i})\}$ be a list of overlapping face pairs (f_i, f_j) in $\mathcal{T}$. If $\mathcal{L}$ is empty, then we found a valid net; otherwise we will use $\mathcal{L}$ to determine the foldability likelihood of $\mathcal{T}$. Given a face f of $\mathcal{T}$, let $CC(f)$ be a set of non-overlapping faces that contains f. We say that $CC(f)$ is *maximized* if no additional faces can be added to $CC(f)$ without overlapping with the members of $CC(f)$. To compute the maximized $CC(f)$, we start with $CC(f) = \{f\}$, and then iteratively test the faces of $\mathcal{T}$ adjacent to the current $CC(f)$ in breadth-first search manner, i.e., only expand $CC(f)$ via the fold edges and not the cut edges. Let f' be an adjacent facet that does not overlap with the facets of $CC(f)$, then $CC(f) = CC(f) \cup f'$.

After the maximized $CC(f)$ is found, the foldability matrix $\mathcal{M}$ is updated so that all elements between f and $f' \in CC(f)$ are increased by one, i.e. $\mathcal{M}(f, f') = \mathcal{M}(f', f) = \mathcal{M}(f, f') + 1, \forall f' \in CC(f)$. This operation is repeated for all faces for a given $\mathcal{T}$.

When multiple unfoldings are performed on the same mesh, the foldability likelihood matrix $\mathcal{M}$ accumulates the likelihood estimation.

4.2. Segment

After the foldability likelihood of all pairs of faces is determined, we use spectral clustering [18] to cluster the faces. Although other clustering methods can also be used, we found that spectral clustering gives consistent and better results. For example, we attempted to use Lloyd's algorithm [19] that ensures all clusters are made of connected faces, but this approach usually results in sub-optimal clusters because the idea of cluster center cannot be easily defined. An example of the clustered foldability matrix using spectral clustering can be found in Fig. 7.

Since we use Steepest Edge Unfolding for determining the foldability which has a strong preference for convex shapes, the segmentations generated by spectral clustering are usually nearly convex (see Fig. 12) which gives us several advantages: 1) each component is easier to unfold to a net. Sometimes the net can

be obtained by using heuristic methods alone (no GA involved, detailed in Section 4.3); 2) it is much easier to find a continuous folding motion for nearly convex shape, e.g., the folding path is a straight line in the configuration space [3], that ensures that the net can be continuously folded to 3D without self-intersection which is critical for building self-folding robot with rigid materials. 3) easier to fold the net by hand as stated in Section 5.2; over existing non-convex unfolders.

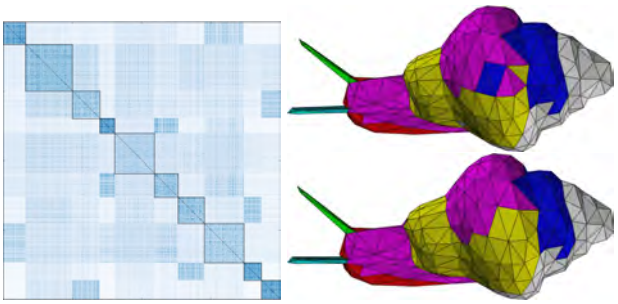


Figure 7: **Left**: Clustered foldability matrix of the monkey model in Fig. 9. A darker pixel indicates higher possibility of unfolding the corresponding face pair without overlapping. 10 blocks along the diagonal represent the 10 clusters found and correspond to the segmentation of the monkey model. **Right**: Before (top) and after (bottom) isolated facets are reassigned.

Spectral clustering does not consider face adjacency, therefore some triangles may be separated from the main components. Our experiments show that, if there exist multiple connected components in a given cluster, these isolated components are much smaller in size than the main component in the given cluster. Even on more complex models as shown in Fig. 12, the isolated faces are usually less than 1% among all faces. Therefore, a post-processing step is suffice to enforce every cluster to contain only connected facets. Given that f is a face disconnected from the the largest component in its cluster C_i and is adjacent to the clusters $\{C_{j\neq i}\}$. Then f is reassigned to a new cluster C_j to maximize the co-foldability

$$\arg\max_j \sum_{f'\in C_j} \text{area}(f')\mathcal{M}(f,f') \ . \tag{1}$$

The snail model shown in Fig. 7 illustrates an example before and after the isolated components are reassigned.

4.3. Unfold to a Single Net

Elliptic vertices have zero or positive Gaussian curvature and have the sum of the vertex angles of adjacent faces $S_v \leq 2\pi$ while *hyperbolic vertices* have negative Gaussian curvature and $S_v > 2\pi$. A convex polyhedron only contains elliptic vertices, one cut on its adjacent edges is sufficient to unfold that vertex without local overlaps, i.e., all the adjacent faces of that vertex do not overlap. For hyperbolic vertices, at least two cuts are required in order to avoid local overlaps on the unfolding. Non-convex polyhedra usually contain hyperbolic vertices, heuristic methods developed for convex polyhedra which find one best cut edge for each vertex no longer works on non-convex polyhedra.

Theorem 4.1. *For a polyhedral surface M, if M has a net N then $\exists W$, weights of dual edges, such that the minimum spanning tree (MST) of the dual graph of weights W leads to N.*

Proof. Given N, we can construct W as following, set weights to 0 for dual edges kept in N and set weights to 1 for dual edges cut to obtain N. □

Unfolding Evolution Fortunately, after segmentation, each component becomes nearly convex (but not exactly), in this case, aforementioned heuristic methods could generate approximate solutions, that are unfoldings with fewer overlaps (sometimes 0 overlap), which give us a good starting point. Our idea is to use Genetic Algorithm to evolve the unfoldings by mutating the weights on the dual edges to reduce the number of overlaps in the unfoldings and finally find a net with 0 overlaps. A genetic algorithm requires: 1) a genetic representation (gene) of the solution domain. In our case, it will be the weights of all dual edges. 2) a fitness function for evaluating the solution. We evaluate an unfolding (generated by finding a MST of the dual graph, the gene represents the dual edge weights) using the fitness score defined as: $f = -(\lambda_o N_o + \lambda_l N_l)$, where N_o is the number of overlaps in the unfolding and N_l is the number of hyperbolic vertices that cause local overlaps in the unfolding, λ_o and λ_l are their coefficients and were set to 1 and 10 respectively in our experiments. Since local overlaps are harder to resolve than global ones thus they play a more important role in the fitness function.

We generate p unfoldings using randomly picked heuristic methods as the initial population. In each generation, b new unfoldings are generated from randomly picked parents using genetic operators (crossover and mutation) and they are used to replace the worst unfoldings (with lowest fitness) in the population. During the evolution, we are expected to see better and better individuals (with fewer overlaps). Once the fitness becomes zero, we found a net.

If GA fails to find a net within certain generations (maybe due to local minima), we restart the evolution for at most r times. If still no net can be found, it means that mesh is likely to be ununfoldable, we will segment it using the proposed method and find a net for each segmentation recursively.

It is important to note that the GA method described above can be replaced with any nonconvex mesh unfolders, including [12]. Our goal here is to demonstrate that a nonconvex mesh unfolders can be greatly enhanced by incorporating it with the clustered foldability matrix. For example, even for a small mesh, such as the Snail model (Fig. 7), the proposed method is three times faster (about 250 seconds) than using GA alone to unfold the entire mesh.

4.4. Close the Loop - Continuous Folding

All previous methods export nets as the final product, however, the loop is still open. Whether there exists a continuous folding

motion that transforms the net back to the original mesh is unknown. It becomes critical if we would like to build a physical self-folding robot with rigid material, for which, bending on the panels is not allowed. We employ the method from [3] to plan continuous folding motion for the net to close the loop. The folding motion found can be either used as a visual guidance for paper crafting or part of folding strategy for the self-folding robot. If it failed to find a folding motion of a net for certain amount of time which means that the net is hard to fold. In this case, we could further segment the mesh/patch which the net is unfolded from. The folding motion can be best visualized from the accompanied video.

5. Experiments and Results

5.1. Setup

The proposed method is implemented in single-threaded C++. Unless stated otherwise, the experimental results reported in this section are setup as follows. To train the foldability matrix, we run the steepest-edge heuristic 400 times. GA population is set to 500 and the maximum GA generation is 2000. In each generation, 40 new unfoldings will be generated. All data are collected on a workstation with two 2.3GHz Intel Xeon E5-2630 CPUs.

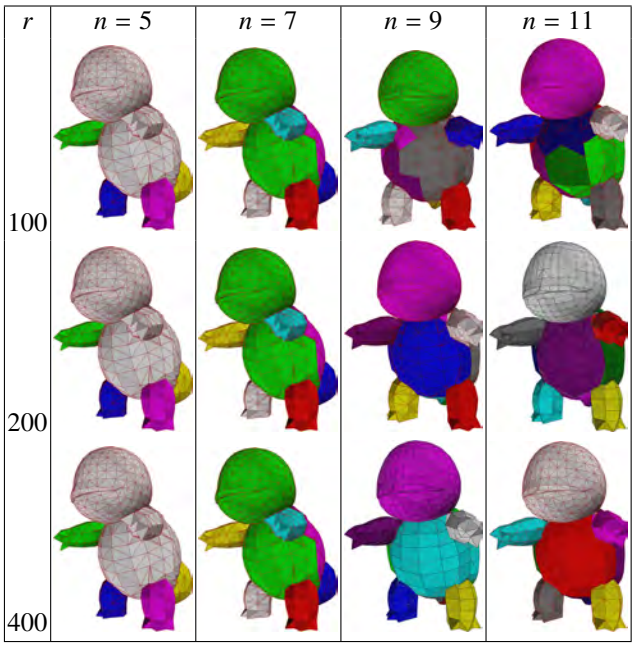
5.2. Segmentation, Unfolding, and Fabrication Results

Figs. 8 to 10 show the segmentations and nets produced by the proposed method and the crafted paper models. All nets are arranged and cut from 12 inch by 12 inch paper sheets; tabs are not used to allow larger nets. The paper craft is then fabricated by gluing the cut edges from inside the model using hot glue gun. One significant crafting benefit provided by the proposed method is that each part is much more convex than the input model, thus the user can simply crease all edges as mountain folds without even consulting fold assignment in the nets, and then only reverse the folds for (usually much fewer) concave edges.

It is worth noting that the difficulty of making these triceratops, monkey and Yi Sun-sin paper crafts in Figs. 8 to 10 is quite high. They took us about 12, 21 and 36 hours to make, respectively. These meshes have not only many triangles (between 1000 and 3000) but also have even none-zero genus. Comparatively, the models studied in the literature are all significantly smaller than ours. For example, two models with 270 faces and 200 faces are crafted by [4]. The most complex model in [10] has 347 faces and requires 25 hours of crafting time . In [12], the most complex model studied has 950 faces and the most complex paper craft created has 344 faces and required about an hour of crafting time. The significant crafting-time reduction in [12], from 25 hours reported in [10] to 1 hour for models with similar complexity, is mostly due to the crafting method. In [12], taping, instead of gluing, is used to connect cut edges. When taping is used to produce a paper craft with 128 faces produced by [10], the reported difference is only 8 minutes at most.

Parameter study There are two main parameters in the proposed method, namely the number of training runs r and the (initial) net (or part) counts n. Table 1 shows a study of various combination of r and n of the Squirtle model. Let us look at the variable r first. When the number of n is small (5 or 7), then 100 runs of training are enough to stabilize the results as the segmentation remains the same when r increases to 200 and 400. However, when the number of n is larger (9 or 11), 100 runs of training are insufficient. The segmentation around the torso becomes unnatural. This unnaturalness disappears when r increases to 200. When r increases to 400, the segmentation changes only slightly. Note that, when $r = 400$, the proposed method produces pretty reasonable results regardless the values of n.

Table 1: Segmentations created by varying the cluster size n and the number of training runs r.



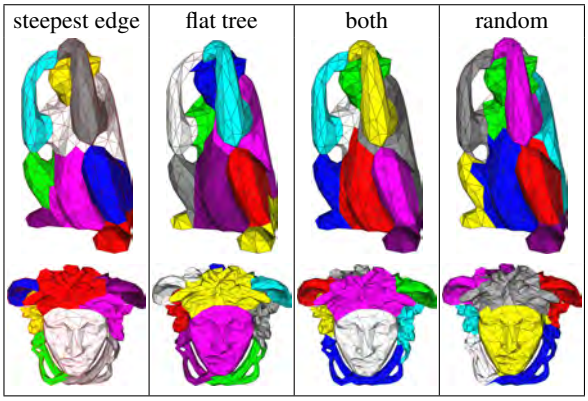
Unfolding heuristics Although we have been using steepest-edge unfolding to create the nets in our examples so far, the training of foldability matrix can be also done using other heuristics. Table 2 shows example outputs of two models using various heuristics including a hybrid heuristic that combines steepest-edge and flat-tree. For all experiments reported in Table 2, the number r of unfolding training runs is fixed at 400 and the number n of segmentation is at 9. First, all segmentation results are similar regardless the heuristics used. The segmentations trained using steepest-edge and flat-tree unfoldings are not identical but (semantically or anatomically) reasonable in their own ways. Even when unfolding using random edge weights is used, the proposed method is able to produce meaningful segmentation (hands of the monkey) although its segmentation boundary is not as clean as the other methods. For instance, the head and torso of the monkey are connected and the face of the

(a) Segmentation (b) Crafted paper model

Figure 8: **Left**: the model of triceratops (1000 triangles) is decomposed into 11 clusters by the proposed method. **Right**: The paper craft of the triceratops model, which is 32 cm wide, 9 cm deep and 15.5 cm tall.

man is not smoothly separated from the hairs. However, it is to our surprise that even randomly unfolding the mesh (followed by overlapping analysis) can provide useful part information.

Table 2: Segmentation results with various training runs. From left to right: (1) 400 runs of steepest-edge unfolding, (2) 400 runs of flat-tree unfolding, (3) 200 runs of steepest-edge + 200 runs of flat-tree unfolding, and (4) 400 runs of random unfolding.



5.3. Compare to Existing Unfolders

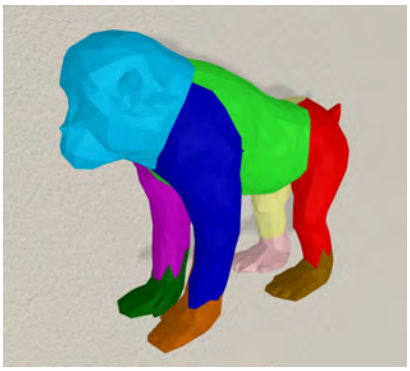
In this section, we compare the proposed method with the state-of-art methods, namely a widely used Pepakura Designer [20] and a genetic programming approach [12]. Table 3 shows the number of nets produced by these methods. The method by [12] is parameterized to minimize the final number of nets, thus can be viewed as a reasonable lower bound provided by a global optimization method. Two numbers are reported by our method in Table 3. The numbers in the parenthesis are the initial sizes provided by the user, and the other number is the number of nets produced. The users usually offer the initial size by counting the number of meaningful parts of a given model. Because not all the parts are unfoldable, our method must produce more nets than requested. Also because of the same reason, the number of the nets produced by our method for models with lower triangle count (such as snail and triceratops) is often larger than those produced by [12]. However, for models with higher triangle counts (e.g. those with more than 2000 triangles), our method often creates fewer number of nets (except the dragon model).

Table 3: Compare the number of components created by the proposed methods, Pepakura Designer 3 [20], and Takahashi et al. [12]. The numbers in the parenthesis reported by the proposed method are the initial sizes provided by the user by counting the number of meaningful parts of a given model.

Models	Triangle size	Number of components		
		Ours	[20]	[12]
Snail (Fig. 7)	574	5 (5)	5	1
Peekaboo (Fig. 13)	1000	9 (9)	18	N/A
Dinosaur (Fig. 13)	1000	9 (9)	29	N/A
Triceratops (Fig. 8)	1000	11 (11)	36	5
Squirtle (Fig. 6)	1718	6 (6)	18	4
Angel (Fig. 13)	2000	16 (11)	42	14
Monkey (Fig. 9)	2000	10 (10)	50	13
Children (Fig. 2)	3000	16 (13)	68	17
Yi Sun-sin (Fig. 10)	3000	11 (9)	79	N/A
Dragon (Fig. 13)	5000	34 (11)	130	16

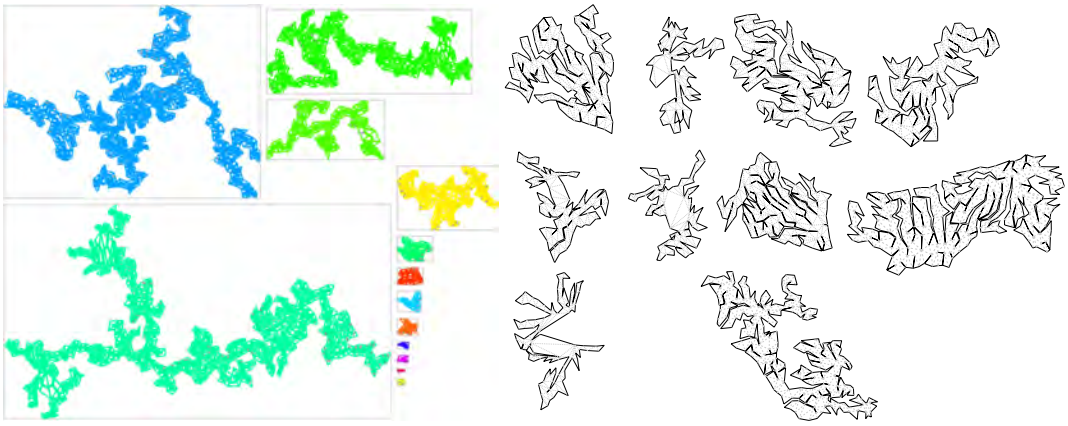
5.4. Compare to Shape Segmentation Benchmark

We also validate the proposed method on some models from the Princeton Shape Segmentation Benchmark [21]. Although we are not using any surface or volumetric shape features, we show that segmenting via foldability analysis achieves competitive results comparing to the state of art segmentation methods. For fair comparison, we set the number of clusters k for each category as the average number of components for that category reported in [16]. Note, measurements can be improved if we choose the best k for each model individually. In Fig. 11, we show rand index (RI) [22], a metric that measures the similarity of whether a pair of faces are in the same segmentation, and cut discrepancy (CD) [21], which is the sum of distances between points in a cut to the closest cut in the ground truth. In Fig. 12, we show the segmentation results obtained by the proposed method.



(a) Segmentation

(b) Crafted paper model



(c) Results from Takahashi et al. Takahashi et al. [12]

Figure 9: **Top left:** The model of monkey (2000 triangles) is decomposed into 10 clusters by the proposed method. **Top right:** The paper craft of the money model, 27 cm wide, 8 cm deep and 23 cm tall. **Bottom left:** The nets generated by Takahashi et al. [12] (13 parts) that is trying to minimize the number of parts. Notice the large variation in net sizes. **Bottom right:** The nets created by the proposed method (10 parts) have even sizes.

6. Discussion and Limitations

The proposed method is capable of producing small number of nets that resemble the semantic parts of the input model. However, it has three main limitations. First, in our current implementation, the initial number of parts remains a user parameter. Even though providing this number is typically not a great burden to the user, it is desirable to use one of the existing methods to estimate the number of semantic parts.

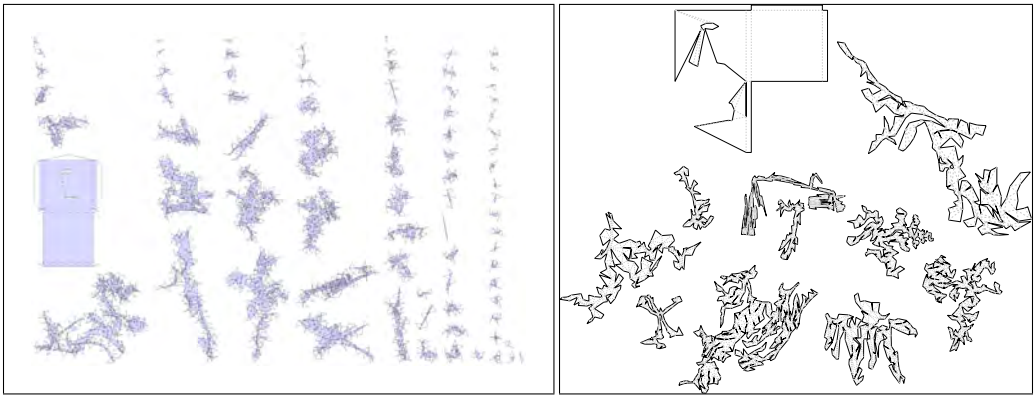
The second limitation of our method is the computation time. Pepakura designer [20] takes between 1 to 6 seconds to unfold the models in this paper. However, it takes about half an hour for our method to generate nets for a mesh of 1000 faces. The running time of our method grows nearly as a quadric function of $|F|$ (if we have a constant maximum generations limit in GA). One way to speed up the computation is to employ graph cut to segment the mesh similar to what shape diameter function [23] and continuous visibility feature [24] did. Note that, while the method from Takahashi et al. [12] is fast for smaller meshes, e.g., 15 seconds for a mesh with 312 faces. but it takes more than 2 hours to finish unfolding a mesh with 2000 faces.

Another limitation comes from the fabrication aspect. Even with the help of semantics and convexity, crafting a paper model with more than 3000 triangles remains time consuming and labor intensive. Models smaller than 3000 triangles, such as most of the models that we showed in this paper, cannot provide much surface details (such as wrinkles or surface texture). It requires further research to find out how these detailed surface features can be preserved without significantly increasing the fabrication burden.

7. Conclusion

With the advances in active materials, making 3D shapes from planer shapes finds its new application: self-folding machines. "Unfolding and folding" is one the prevailing ways to design and build such robots. In this paper, we propose a novel approach that unfolds and segments a given 3D mesh simultaneously by learning from self-unfolding. Compare to existing methods, the proposed method provides semantic segmentations, reduces the number of components and makes folding much easier. We show that the nets produced by the proposed method can be used to create more complex paper craft (e.g., 3000 faces in 36 hours in Fig. 10) comparing to the results reported in the literature (e.g., 347 faces in 25 hours [10]).

(a) Segmentation (b) Crafted paper model



(c) Results from Pepakura Designer [20]

Figure 10: **Top left:** The statue of Korean general Yi Sun-sin (3000 triangles) is decomposed into 11 clusters by the proposed method. **Top right:** The paper craft of the model, which is 9.7 cm wide, 9.7 cm deep and 21.2 cm tall. **Bottom left:** The nets generated by Pepakura Designer [20] (79 parts). **Bottom right:** The nets generated by the proposed method (11 parts). The unfolder developed by Takahashi et al. [12] is unable to unfold the model perhaps due to that the mesh is not water tight.

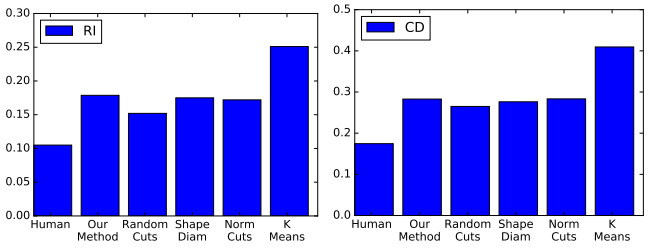


Figure 11: Comparison of rand index and cut discrepancy measures on Princeton Shape Segmentation Benchmark. For both RI and CD measures, the lower the better.

Acknowledgement

This work was supported in part by US NSF EFRI-1240459, IIS-096053, CNS-1205260, and AFOSR FA9550-12-1-0238. J.-M. Lien were also supported by the Korean Federation of Science and Technology Societies (KOFST) grant funded by the Korean government. Y.-J. Kim and Y.-H. Kim were supported in part by NRF in Korea (2014K1A3A1A17073365, 2015R1A2A1A15055470).

References

[1] Dougherty D. The maker movement. innovations 2012;7(3):11–4.

[2] Felton S, Tolley M, Demaine E, Rus D, Wood R. A method for building self-folding machines. Science 2014;345(6197):644–6.

[3] Xi Z, Lien JM. Continuous unfolding of polyhedra - a motion planning approach. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Hamburg, Germany; 2015, p. 3249 –54.

[4] Mitani J, Suzuki H. Making papercraft toys from meshes using strip-based approximate unfolding. In: ACM Transactions on Graphics (TOG); vol. 23. ACM; 2004, p. 259–63.

[5] Massarwi F, Gotsman C, Elber G. Papercraft models using generalized cylinders. In: Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on. IEEE; 2007, p. 148–57.

[6] Schlickenrieder W. Nets of polyhedra. Master's thesis; Technische Universität Berlin; 1997.

[7] Shatz I, Tal A, Leifman G. Paper craft models from meshes. The Visual Computer 2006;22(9-11):825–34.

[8] Katz S, Tal A. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans Graph 2003;22(3):954–61. doi:http://doi.acm.org/10.1145/882262.882369.

[9] Julius D, Kraevoy V, Sheffer A. D-charts: Quasi-developable mesh segmentation. In: Computer Graphics Forum; vol. 24. Wiley Online Library; 2005, p. 581–90.

[10] Straub R, Prautzsch H. Creating optimized cut-out sheets for paper models from meshes. Tech. Rep.; Karlsruhe Institute of Technology; 2011.

[11] Glover L. KitRex. 2014. URL: http://www.kit-rex.com/.

Figure 12: Example results of the proposed method on Princeton Shape-Segmentation Benchmark models.

[12] Takahashi S, Wu HY, Saw SH, Lin CC, Yen HC. Optimized topological surgery for unfolding 3d meshes. In: Computer Graphics Forum; vol. 30. Wiley Online Library; 2011, p. 2077–86.

[13] O'Rourke J. Unfolding polyhedra. 2008.

[14] Lien JM, Amato NM. Approximate convex decomposition of polyhedra. In: SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling. New York, NY, USA: ACM Press. ISBN 978-1-59593-666-0; 2007, p. 121–31. URL: `http://doi.acm.org/10.1145/1236246.1236265`. doi:`10.1145/1236246.1236265`.

[15] Asafi S, Goren A, Cohen-Or D. Weak convex decomposition by lines-of-sight. In: Computer Graphics Forum; vol. 32. Wiley Online Library; 2013, p. 23–31.

[16] Liu G, Xi Z, Lien JM. Nearly convex segmentation of polyhedra through convex ridge separation. In: Symposium on Solid & Physical Modeling (SPM); also appears in Journal of Computer-Aided Design. 2016,.

[17] Lucier B. Unfolding and reconstructing polyhedra. Master's thesis; University of Waterloo; 2006.

[18] Ng AY, Jordan MI, Weiss Y, et al. On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems 2002;2:849–56.

[19] Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY. An efficient k-means clustering algorithm: Analysis and implementation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 2002;24(7):881–92.

[20] Tama Software Ltd . Pepakura designer. 2016. URL: `http://www.tamasoft.co.jp/pepakura-en/`.

[21] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3d mesh segmentation. ACM Transactions on Graphics (TOG) 2009;28(3):73–.

[22] Rand WM. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association 1971;66(336):846–50.

[23] Shapira L, Shamir A, Cohen-Or D. Consistent mesh partitioning and skeletonisation using the shape diameter function. The Visual Computer 2008;24(4):249–59.

[24] Liu G, Gingold Y, Lien JM. Continuous visibility feature. In: 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA: IEEE; 2015, p. 1182 –90.

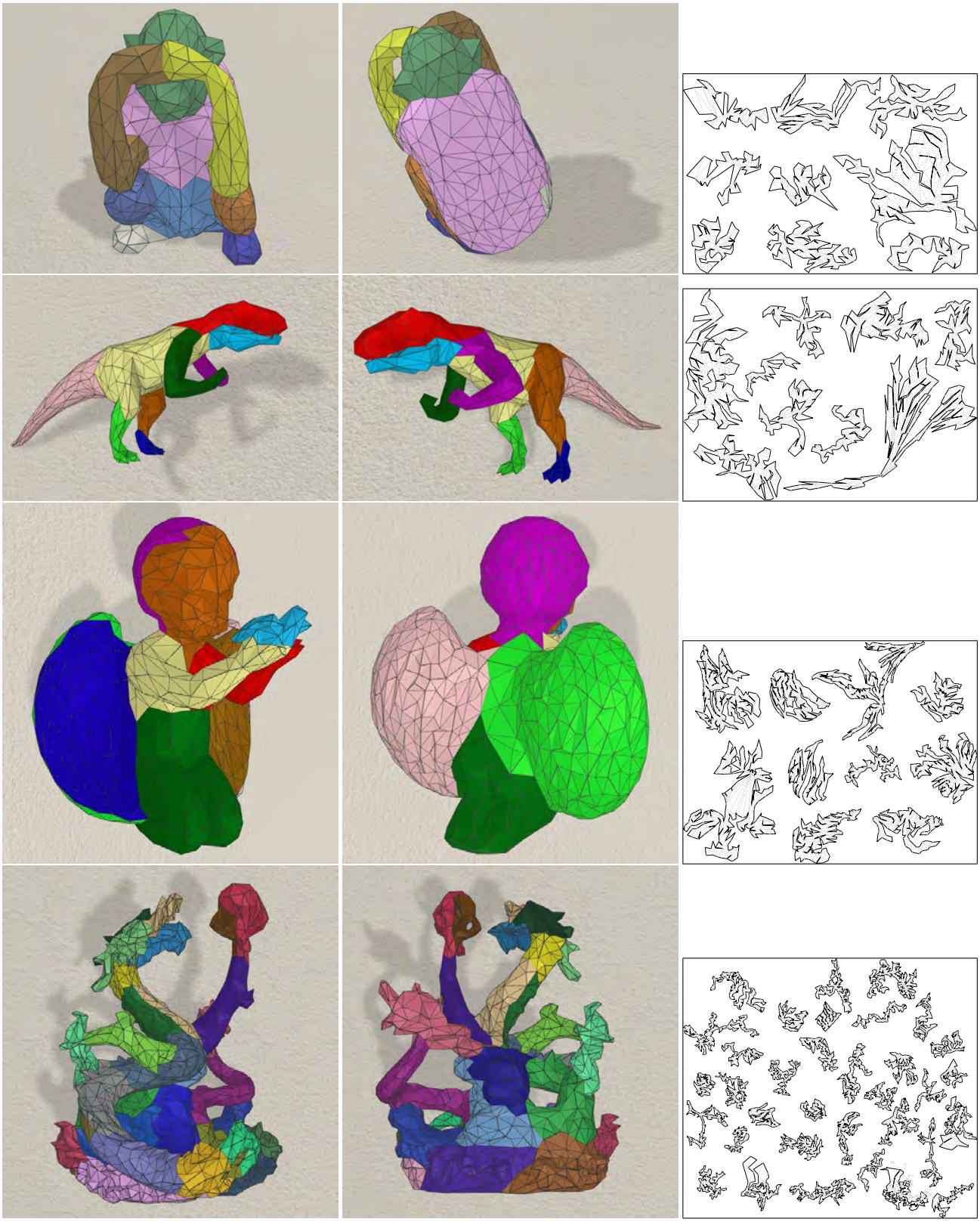


Figure 13: Additional examples. From top to bottom: Peekaboo, dinosaur, angle, dragon. See Table 3 for triangle count and net count.