Cover-it: An Interactive System for Covering 3D Prints

Ali Mahdavi-Amiri* Philip Whittingham† Faramarz Samavati‡

University of Calgary

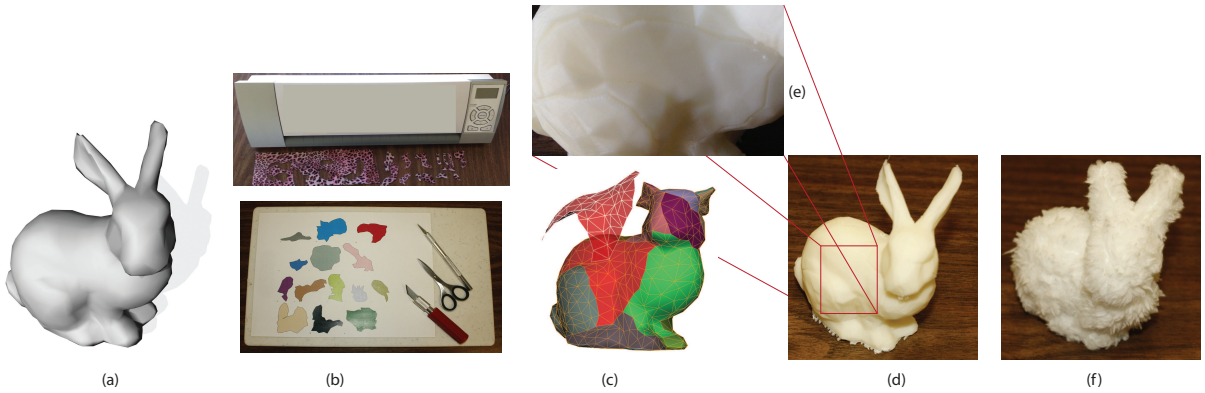

Figure 1: The work-flow of Cover-it: Starting with (a) an input model, we generate (b) a 2D layout of simple patches and (c) an animation showing how these patches fit on (d) the 3D print of the model. As guides, (e) the patch outlines are etched into the print to support the creation of (f) the final covered print.

Abstract

The ubiquity of 3D printers has made it possible to print various types of objects, from toys to mechanical objects. However, most available 3D printers are single or double colors. Even printers that can produce objects with multiple colors do not offer the ability to cover the object with a desired material, such as a piece of cloth or fur. In this paper, we propose a system that produces simple 2D patches that can be used as a reference for cutting material to cover the 3D printed object. The system allows for user interactions to correct and modify the patches, and provides guidelines on how to wrap the printed object via small curves illustrating the patch boundaries etched on the printed object as well as an animation showing how the 2D patches should be folded together. To avoid wasting materials, a heuristics method is also employed to pack 2D patches in the layout. To compensate the effect of inflation resulted from covering objects with thick materials, an offsetting tool is provided in Cover-it. In addition, since many low scale details of an object is not visible after covering, a mesh can be simplified in Cover-it to reduce the number of 2D patches.

Index Terms: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages; I.3.8 [Computer Graphics]: Applications

1 Introduction

Nowadays, numerous inexpensive 3D printers are available for fabricating 3D objects. This has generated exciting opportunities for creative applications and stimulated both researchers and industrial companies to explore the use of the 3D printing technology. As a result, many different models — such as prototypes, toys, mechanical objects and accessories — are printed and utilized today. However, most current 3D printers are only able to print objects in one or two colors and only in a single material. More advanced (and expensive) 3D printing technologies can produce objects in multiple colors. However, these objects are not able to be produced with various coverings, such as fur, leather, decorative paper/fabric, or other textiles, which is necessary in the design of commonly available objects. To address this, we can use 3D printers to generate the underlying 3D model, and then cover them manually by the desired coverings. However, appropriately wrapping a complicated 3D object with a (2D) material is not an easy task.

In this paper, we present *Cover-it*, an interactive system specially designed for guiding the task of wrapping 3D prints with various coverings. For a given 3D object, Cover-it creates a 2D layout of simple patches for covering 3D prints (see Fig 1). This layout can be used as a stencil guide for cutting patches from various coverings. Cutting can be done manually or with the aid of digital cutter printers (see Fig 1 (b)). To keep the process of attaching 2D patches on 3D prints simple, the number of patches should be small and, each patch should have simple boundaries. Furthermore, in the interest of pasting each patch on the object without wrinkles, the patch should be developable.

These criteria have been previously conceived in applications such as paper craft and stuffed toy creation [21, 8]. As a result, we use D-Charts [8] which is a simple and efficient segmentation method. To customize the result, we provide a set of interactions to locate segments and seams.

Although D-Charts can successfully segment a given mesh to a set of developable patches, there are other aspects specific to 3D print covering in addition to developability of patches. In general, two important constraints exist: Patches must be accurately attached together (patch-patch alignment). Patches must be accurately attached to their corresponding regions on the 3D prints (patch-surface alignment). The second condition does not exist for paper craft and stuffed toy applications. However, to cover 3D prints, the second condition is very important as a slight violation

*e-mail: amahdavi@ucalgary.ca
†e-mail: philip.whittingham@gmail.com
‡e-mail: samavati@ucalgary.ca

in the position or orientation of the patch over the 3D print leads to uncovered and/or noticeable overlaps. Pasting patches on 3D prints with respect to the second constraint can be an extremely tedious and time consuming task. For example, the process of pasting the red patch in Fig 1 (b) on the bunny (Fig 1 (d)), even with the use of the color-coded diagram in Fig 1 (c), is hard and unreliable. As a result, it is necessary to provide a clear guide for the pasting process. In addition, the seams between patches present an important consideration for the covering problem. For instance, they should be ideally moved to less visible regions or they can be aligned with feature curves to respect an important aspect of the surface. Therefore, the ability to interactively control the cutting layout is desirable.

In Cover-it, some guidelines are also provided to make the process of attaching patches to the 3D prints easier. Color-coding the patches is beneficial but not enough, as it remains difficult to locate the patch accurately on the 3D print. As a result, in Cover-it, the boundary of each patch is carved as a small but visible etch on the 3D print (see Figures 1 (e) and 8). Therefore, the exact location of 2D patches on the mesh is apparent. In addition, as evident in Fig 1 (c), it is hard to determine the final orientation of each of these 2D patches. Therefore, in Cover-it, an animation is provided showing how the 2D patch is oriented on the 3D print.

An additional set of user interactions are also designed in Cover-it to customize the final result of the system. In Cover-it, it is possible to simplify the mesh to decrease the number of segments. This simplification technique works based on a developability factor for creating objects at different resolutions. In fact, the mesh is simplified at vertices with a low Gaussian curvature and small incident triangles so that the overall shape of the mesh is affected as little as possible. If a mesh is covered with a thick material, the final object is undesirably inflated. To compensate this artifact, an offsetting tool is provided in Cover-it to make an offset surface corresponding to the thickness of the employed material. 2D patches can be created using a cutter printer that receives the 2D layout as an input and cuts the 2D patches out of a resource material. To avoid wasting materials, the 2D layout resulting from the segmentation is also packed using a heuristic approach [1]. In Cover-it, it is also possible to control the position of 2D patches on the layout using simple rotation and translation tools.

In summary, this paper makes the following contributions:

- The proposed solution of wrapping 3D prints with 2D coverings to provide more interesting and realistic objects.
- Analysis of the requirements for covering 3D prints.
- The design of a system that provides a low number of developable patches with simple boundaries packed in a 2D layout in addition to pasting guidelines such as an animation and patch boundaries carved on the object. Other useful interactions, such as control over the placement of seeds and seams, are also included.

We organize the paper as follows: Related work is presented in Section 2. The requirements of a proper covering of 3D prints is analyzed in Section 3. Section 4 provides an overview of the segmentation and packing techniques employed in Cover-it. Guidelines that are provided in Cover-it to simplify the pasting process are described in Section 5. We present the user interactions of Cover-it in Section 6. Section 7 presents the results and discussion and we finally conclude in Section 8.

2 Related Work

Cover-it is a system that provides a set of 2D patches to wrap a 3D print covered with a set of developable patches. Therefore, the related work of our system includes previous work in model fabrication, developable surfaces, paper crafts, and segmentation. In Cover-it, the corresponding region of each patch on the surface is highlighted by carving the boundary of the patch on the 3D print. As a result, our work is weakly related to 3D puzzle creation. In the following subsections, we discuss the previous work of each category in detail. Naturally, the scope of some of these works spans multiple categories.

2.1 Stuffed Toys and Model Fabrication

The final product of our system is a 3D print covered with a given material, which is similar to plush toys except that, in our system, a 3D print underlies the outer material rather than a soft stuffing. Plushie is an interactive system for making toys using a sketch-based interface [17]. However, whereas Plushie is a modeling system that allows a user to model a toy, our system accepts any (pre-existing) 3D model as an input mesh. In fact, 3D model in Plushie is not accurately respected. Similar to Plushie, Pillow is designed to make toys using an interactive system [6]. Skouras et al. [24] also propose an interactive system for inflatable models that are defined by a set of drawn seams. These systems are tools for creating objects and do not handle existing 3D models, unlike the work of [7], which considers the problem of covering pre-existing 3D objects. However, the problem tackled by the work is not the covering of the surface of the model. Rather, the purpose of [7] is to design a cover for personal belongings to protect them from dust and damage.

Cover-it also shares similarities with systems designed for model fabrication. Wang [30] propose a method to interpolate boundary curves using stretch-free patches for prototyping leather patches on objects such as shoes. In [29], a system is designed to provide a 2D patterns that can be sewn together to provide a clothing or a stuffed toy. Chen et al. [4] propose an interesting system to fabricate a wooden 3D model. The motivation for their work, regarding the lack of simple methods to print an object covered with different materials and in various colors, is the same as ours. In their work, the purpose is to provide a small number of solid pieces to be assembled, therefore they significantly simplify the object. However, we intend to work with soft coverings wrapped on 3D prints and can support more detailed objects. In addition, there are other aspects for covering 3D prints such as aligning the patches on a 3D surface (i.e. patch-surface alignment), which is not the case for [4] as there is no underlying 3D print.

In D-Charts [8], the developable segments of the mesh are detected using a region growing algorithm that considers developability, compactness, and smoothness of the boundary edges as the elements of the fitness function. The purpose of their segmentation is to produce stuffed toys using soft materials. To guide the correspondence between 3D patches and 2D patches, they use a color coding for the patches. Since they make stuffed toys with no underlying 3D print, patch-patch alignment is a sufficient constraint. The segmentation component of our system is very similar to [8]. In our proposed system, patch-surface alignment is also considered, as it significantly affects the final results. To fulfill this constraint, a set of interactions and guidelines are provided.

2.2 Developable Surfaces

Detecting the developable patches of a given surface has important applications in computer graphics. Developable surfaces are traditionally defined as surfaces with zero Gaussian curvature. Gaussian curvature is the product of the principal curvatures, κ_1 and κ_2, and can be used as a metric to find developable patches.

To calculate the Gaussian curvature, Yamauchi et al. [32] use an area of the Gauss map and apply it to identify the developable patches of a surface. In [18], a sketch-based modeling approach is proposed to produce developable surfaces by interpolating the boundary curves using a developable triangulation. Liu et al. [13] propose a method to produce conical and circular meshes using an

optimized subdivision on quad meshes with planar faces. Kilian et al. [9] propose an interesting system (curved folding) for designing developable surfaces in which a model can be approximated by a single planar sheet of material without stretching, tearing or cutting. The work of Tachi [27] presents a different method to fold a given mesh shape using several foldings without cuts.

2.3 Paper Crafts

Converting a mesh to a paper craft, which involves unfolding a given mesh to a set of foldable paper patches is also related to our work. Mitani and Suzuki [16] propose a method in which the given mesh is initially partitioned along its features lines. Partitions of the mesh are then segmented to smaller partitions in such a way that these segments can be approximated by triangle strips. They then use the triangle strips to cut the paper and construct the paper craft. This method drastically simplifies the object and the resulting triangle strips are narrow and hard to attach.

In [21], a given mesh is segmented into a relatively small set of patches with smooth boundaries via an optimization method that locally fits a conic or planar surface to the mesh. In [15], the same problem is tackled by approximating the patches using a set of generalized cylinders. Although these methods preserve the overall look of the mesh, some undesirable pointy patches are created.

Unfolding the mesh into large triangle strips is also discussed in [26, 28]. Takahashi et al. [28] generate a guide to fold the paper craft using color coding along the cuts. However, their final result is still composed of a complicated patch that is hard to fold. Our work, by contrast, is not limited only to paper and can accept a wide variety of covering materials with different properties. As with the aforementioned model fabrication methods, paper crafts are only constrained to patch-patch alignment, as opposed to our method in which patch to surface alignment is important.

2.4 Segmentation

There exist many previous works that segment a given mesh to a number of patches for different applications such as texture mapping, remeshing, morphing, multiresolution and more [20]. Segmentation is the problem of partitioning a given mesh to a number of smaller patches with respect to one or more objectives. These objectives can be measured in terms of the planarity of the patches, human perception, or a topological constraint. In our proposed system, we look for 2D patches that can be attached together on top of 3D prints. Therefore, the developability of the segment is one of the objectives.

Sander et al. [19] use the planarity of patches as a metric to segment a mesh into an atlas of geometry images. In [5], a new metric for planarity called $\ell^{2,1}$ is defined, which is an ℓ^2 measurement of normals. Using this metric, a remeshing technique is proposed that clusters faces with close normal vectors. To identify planar segments of mechanical objects, a variational mesh segmentation has been proposed by Yan et al. [33] in which a quadratic function fitting is used to estimate the planarity of the object with error metrics similar to those offered by Cohen-Steiner et al. [5]. In order to morph polyhedral surfaces, a decomposition of a model to a number of patches is proposed by Shlafman et al. [23] by considering a linear combination of the physical and angular distances between faces. Physical distance is estimated by the distance between the barycenter of the faces and angular distances are given by the dihedral angles between the faces. Although these segmentation methods perform very well in identifying segments that satisfy their objectives, they do not provide a complete system for our purpose of finding patches that simplify the pasting process to 3D prints.

2.5 3D Puzzles

As finding the correspondence between 2D patches and 3D patches on the surface of the 3D print can be challenging, working with Cover-it can be similar to solving a 3D puzzle. In 3D puzzles, a mesh is also segmented into several pieces in order to provide a challenging and fun experience. Lo et al. [14] generate a 3D polyomino puzzle in which a 3D object is formed of small pieces that connect to each other at the boundaries. Xin et al. [31] create a 3D burr puzzle in which all the pieces except one stay stationary. Song et al. [25] define an interlocking puzzle by subdividing the volume enclosing a 3D object. Although these models segment a mesh to a set of pieces, their intention is different from Cover-it. Cover-it tries to provide a simple fabrication for covered 3D prints by providing guide lines as opposed to puzzles, which should be challenging by nature.

3 Requirements Analysis

In this section, we describe why we have chosen specific methods, guidelines and interactions in Cover-it. The purpose of Cover-it is to wrap the 3D print of a given triangular mesh with different coverings. There exists an associated 3D triangular mesh for each 3D print. One obvious solution is to cut small triangular pieces associated with the triangular faces of the mesh and paste each piece on the 3D print. This can obviously become a very tedious task as a mesh can have several thousand triangular faces. On the other extreme, one can try to cover the entire 3D print using only a single patch, which may have several long cuts. For example, one large triangle strip can be generated, similarly to some works in paper crafts [26, 28]. However, this is also very difficult, as following a long triangle strip on the 3D print and pasting it at the right position is a hard process. Our initial approach generated such a monolithic patch, resulting in the wrapping process for the bunny (See Fig 2) taking several hours. In fact, we did not find any advantage to the single piece approach as lots of effort is already spent on aligning cuts between sub-regions (e.g. 1 and 2 in Fig 2 (c)). Therefore, the covering can include multiple but preferably a small number of patches (Fig 2 (c)). Patches with complex boundaries are also cumbersome to attach, as each requires aligning many edges on the surface and along neighbor patches. This observation is similar to the work on D-Charts proposed by [8].

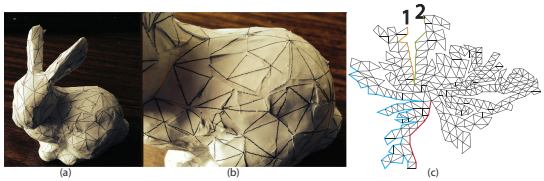

Figure 2: (a) A bunny covered by a paper triangle strips. (b) Gaps are noticeable on the the 3D print. (c) A triangle strip used for covering the bunny. The boundaries are complex and it is difficult to associate them with their patches on the surface. Smooth boundaries (red) are easier than complex boundaries (blue). (1) and (2) should be aligned with each other.

Patches with simple boundaries are better for alignment and attachment. However, our purpose is to paste the patches on the surface of the 3D prints. Our initial experiments showed that it is difficult to paste the patches in the exact position given only a color coded model as a guide (see Fig 3 (c)). Although the virtual model can be rotated and zoomed using standard interactions, guessing the exact orientation of the patches on the surface is difficult. To quantify the difficulty of this task and also observe the issues of pasting, we designed a pilot study and asked 10 people to paste a 2D patch on the 3D print of bunny. As preparation, we located a quadrilateral patch on the bunny for the participants to paste (Fig 3 (a), (b)). We use this quad patch as a reference for measuring and digitizing the error of the pasting process. On the 2D patch, we locate three

reference points whose exact positions on the quad patch were previously captured. After participants placed the patches on the quad patch, we compared the position of the points with their exact positions by aligning quad patches. On average, each participant spent 112 seconds doing this task. Fig 3 (d) shows the distribution of the reference triangles for all 10 cases. The outlier triangle (light blue) belongs to a person who tried to do this task very fast. To define a metric for measuring correspondence error, we used the common areas between the participant's triangle and the reference triangle. For this, we employed a simple Monte Carlo technique to measure overlapping area of these triangles. The study showed that the pasted patch deviated from its exact location by a factor of 60 percent. Such inaccuracy for one piece can create significant misalignment in consecutive pieces. For example, Fig 14 shows the final covering of the bunny by one of the participants in which many parts of the 3D print is not covered properly. The participant spent about 45 minutes to cover the entire 3D print.

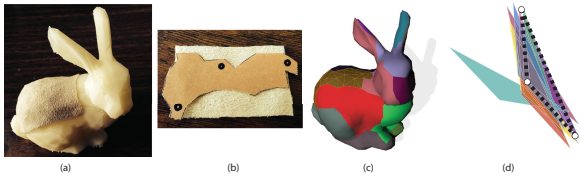

Figure 3: (a) A 3D print and a quadrilateral patch assigned to the surface as a benchmark. (b) Attaching the patch on the benchmark. (c) The 3D object associated with the printed model as a reference for users. (d) Reference points are white vertices and the reference triangle is shown with dashed lines. The reference triangle of each user is drawn in a unique color. Deviation from the reference is noticeable.

To aid the pasting process, we designed two additional tools (in addition to color coding the patches). We provide an animation that shows how a 2D patch corresponds to the patch on the surface. In addition, the boundary of each patch is carved on the 3D model, providing a useful hint to where the patch should be located (see Fig 8).

We also noticed that when a 3D print is covered, depending on the covering material, low scale details are no longer visible. As a result, it is possible to reduce the mesh and obtain simpler patches. In many cases, only a subset of the surface should be covered or the user may require that the seams follow a specific path on the surface as even a simple manipulation and adjustment of seams and patches on the virtual 3D model can reduce the difficulty of the pasting process. For instance, suppose that we wish to dress the Beethoven model in a suit (see Fig 13). Unless interactions are provided to place seams in the proper positions, this task is impossible. Therefore, in Cover-it, seams can be selected and interactively controlled and edited. The positions of patches can be also controlled by selecting a seed triangle from which a patch is grown. These interactions are described in Section 6.

4 Patch Detection

To cover a mesh with a set of material pieces, we need to segment a 3D object to a set of 2D patches. Each 2D patch is associated with a 3D patch on the 3D print. We use segmentation method proposed in D-Charts [8] due to its simplicity, interactivity, and possibility to accept several different factors in its fitness function. Starting from a set of seed triangles and based on the fitness value, a triangular face is attached to the current selections. Using such a method, seams are located along the edges. Although the result of this segmentation method is dependent on the initial location of the seeds and a bad distribution of seeds may result to undesired patches, it can be a useful property since it provides a flexibility for the system to locate patches on the desired locations. This way, the final result is not totally dependent to the optimization technique. In this section, we present an overview of the algorithm that detects patches.

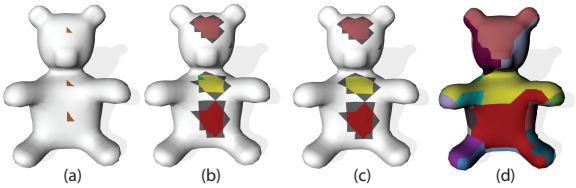

Figure 4: (a) Initial patches are seeded onto the mesh. (b) Patches are grown and the face with the best fitness value (green triangle) is iteratively added to the patch (c). (d) Patches cover all the faces of the mesh.

4.1 Algorithm Overview

The 3D mesh needs to be segmented with respect to the desired properties of the 2D patch layout, which is at least an NP-complete problem and often NP-hard [20]. To simplify this, similar to D-Charts [8], we use a multiple seeds region growing algorithm to make 2D patches. The algorithm initializes the patches using a set of initial seeds that are selected triangular faces. These seeds can be chosen to be geodesically far from each other (using Dijkstra's algorithm). In Cover-it, these seeds can also be interactively chosen if a particular area should be covered without a seam. Among the triangular faces in the neighborhood of a seed, the one with the best fitness value (within an acceptable tolerance) is attached to the seed's patch (see Fig 4).

Patches corresponding to the initial seeds may not cover all the faces of the object, due to the tolerance constraint defined on the fitness of triangles. In this case, more seeds are automatically added to a set of uncovered triangles and the process repeats until all faces of the mesh are covered (see Fig 4 (d)). Additionally, each 3D patch has a corresponding 2D patch to which its triangles are projected, so that the shape (area and angle) of the triangles is preserved as much as possible. It is possible to perform this task using a parameterization technique preserving the shape such as ABF++ [22].

4.2 Fitness Function

For a given triangular mesh M, a fitness function F is used to evaluate whether a face f_a can be attached to patch p_i, which has been initialized by seed s_i. Large fitness values indicate that face f_a should be added to p_i. Function F is a combination of elements, each playing an important role in the formation of the final patches on M. In Cover-it, we define the total fitness function $F = \alpha F_{dev} + \beta F_{comp} + \chi F_{smooth} + \delta F_{vis}$. We can control the effect of each of these elements by varying the coefficients of F. Similar to D-Charts, F_{dev}, F_{comp}, and F_{smooth} are functions to respectively evaluate the developability, compactness, and smoothness of the boundary of patches.

We have added F_{vis} to the fitness function used in Cover-it for pushing the seams to the least visible parts of the 3D print. Therefore, we wish to locate the seams at occluded areas such as beneath the 3D print or its standing regions. Cover-it provides an interaction in which the 3D model can be placed on a plane with a specific pose. Then, the visibility of face f_a is calculated against a defined plane ρ by $F_{vis} = \frac{ACOS(n_{fa} \cdot n_\rho)}{\pi}$. It is possible to define multiple planes in scenarios for which the 3D printed object is intended to be located back to a wall. In the case of multiple planes, F_{vis} with the smallest value is chosen. In Cover-it, in addition to defined planes, it is also possible to use ambient occlusion maps to determine the visibility of faces as a factor in F_{vis} [10].

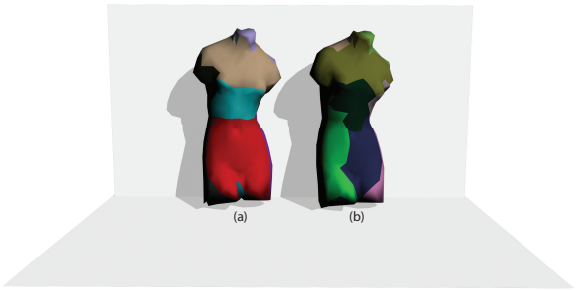

Figure 5: (a) Venus with visibility factor relative to the planes included in the fitness function. (b) Venus without visibility factor. The seams in (a) are pushed further from the front (visible) side of the object than in (b).

4.3 Packing 2D Patches

To avoid wasting materials, it is desired that the patches are packed as much as possible in the 2D layout Ω which represents the domain of the covering layout. 2D patches are extracted from Ω which is typically rectangular. If Ψ is the bounding box encapsulating all 2D patches, waste is defined as $\omega = \Omega/\Psi$. It is desired that the ω be minimized. This problem is known to be NP-Hard which needs an optimization or heuristic technique to estimate the configuration of 2D irregular shapes in which the waste in minimal [3, 2, 1]. In cover-it, we use the heuristic technique proposed in [1] to minimize ω. Following assumptions are made in [1] that are compatible with our need:

- Patches are irregular polygons without holes and Ω is rectangular and large enough to contain all the patches.
- Patches can be rotated and they should not have overlaps while they stay within Ω.

To find the optimal packing, a directed graph with nodes corresponding to all possible allocations of patches is used. A path from an initial node (the allocation of the first patch) to the goal node which is the last allocated patch to the layout with the lowest waste is the solution for the problem. Since possible allocations of patches contain many configurations of rotated patches with different orders, some relaxation rules are used in [1] to shrink the search space. Examples of these rules are to constrain possible rotations for a patch or to prune the graph when a node imposes a large waste to the layout. These relaxation rules may produce good results that are not necessarily optimum. As a result, in Cover-it, we have also provided simple interactions such as rotating and translating 2D patches to edit and improve the 2D layout. Fig 6 illustrates the results of the packing method and some interactions in Cover-it applied on the 2D layout of Furry Wolf illustrated in Fig 16.

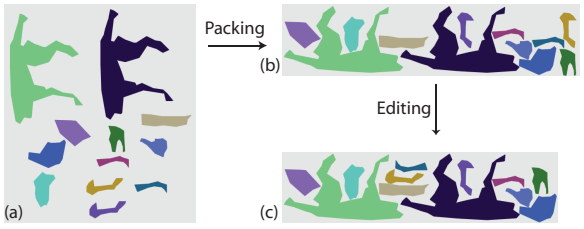

Figure 6: (a) 2D patches of Furry Wolf. (b) Patches are packed in the 2D layout using the heuristic method. (c) Some patches are edited by rotations and translations.

5 Pasting Guides

As concluded in Section 3, it is necessary to have proper guides in the the pasting process. Therefore, Cover-it provides three methods to aid pasting process. In the following, we discuss these methods.

Carving the 3D Print: As discussed in Section 3, we etch the 3D prints along the patch boundaries. To do so, the patches are contracted by slightly moving the boundary edges and the voids between the edges are filled using a triangulation between the two boundaries (see Fig 8). Formally, let vertices v_0 and v_1 forming edge e in triangle T belong to patch p_i (see Fig 7). The new positions of v_0 and v_1 — denoted by w_0 and w_1 — are determined by $w_k = \alpha v_k + (1-\alpha)v_2$ in which $k = 0, 1$ and v_2 is the other vertex of triangle T. e also belongs to another triangle $\acute{T}$ belonging to patch $\acute{p_i}$. v_0 and v_1 also receive two new positions, $\acute{w_0}$ and $\acute{w_1}$ using $\acute{v_2}$, the third vertex of $\acute{T}$. A quadrilateral void $q = (w_0, w_1, \acute{w_1}, \acute{w_0})$ is made due to the repositioning of vertices. To fill q and carve the boundary edge, we introduce two new vertices u_0 and u_1 that are determined by moving v_0 and v_1 in the direction opposite to their normal vectors ($u_i = v_i - \beta n_i$ in which $i = 0, 1$ and n_i is the normal of v_i). Now, eight triangles, as shown in Fig 7, is used to fill the gap (see Fig 7 (d)). This process leads to an etch along the boundary that is visible on the 3D prints. The width and depth of the etches are controllable in Cover-it by changing the value of α and β.

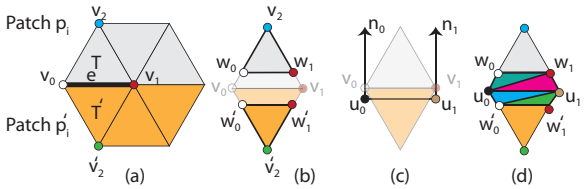

Figure 7: (a) Edge e is in between two triangles T and $\acute{T}$ in two patches p_i and $\acute{p_i}$. (b) Triangles T and $\acute{T}$ are contracted by moving boundary vertices along the edges. (c) v_0 and v_1 —the original vertices of T and $\acute{T}$— are moved along their normal vectors n_0 and n_1. (d) The gap between the contracted triangles is filled by a retriangulation.

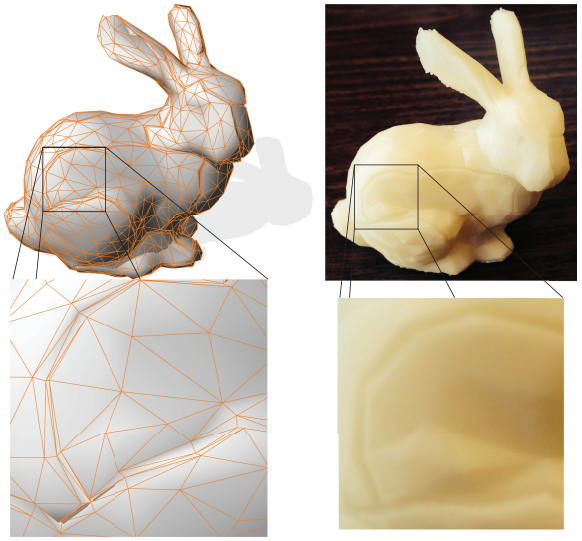

Figure 8: Left: Etches on the 3D model. Right: Etches on the 3D print.

Color Coding: Similar to common segmentation visualizations

[20], we use a color coding in which each patch receives a unique color. We try to establish a strong contrast between the colors of neighboring patches in order to enhance a visual difference between the faces of the patches.

Animation: Animation is a useful tool to provide an insight to the users about the 3D objects [11]. As the third method for guiding the pasting process we use an animation which simulates how the 2D patches q_i should be located on the 3D patches p_i. In the animation, the 2D patches are located near the 3D patches and gradually morphed to their destination position (see Fig 9). To do so, we first estimate the corresponding orientation of q_i relative to p_i. The transformation T that is used to appropriately orient q_i along p_i is estimated by averaging all the transformations T_k that map the faces $f_a \in p_i$ to $g_a \in q_i$ (i.e. $T = \sum_{a=1}^{n} \frac{T_a}{n}$ in which n is the number of faces in p_i). In this way, Tq_i is approximately placed on p_i. Afterwards, Tq_i is translated by αN in which N is the average of the normal vectors n_a of all faces in p_i (i.e. $N = \sum_{k=1}^{n} \frac{n_k}{n}$). $\acute{q}_i = Tq_i + N$ is now located appropriately on top of its corresponding patch p_i. Eventually, to morph the patch $\acute{q}_i$ to p_i, a linear interpolation between the vertices of $\acute{q}_i$ and p_i is used.

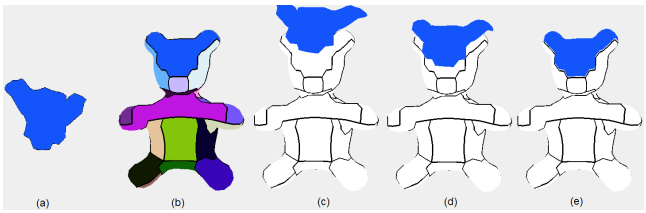

Figure 9: (a) A 2D patch. (b) 3D model visualized in its patches. (c), (d), (e) Sequences of the animation showing how the 2D patch in (a) is morphed to its corresponding 3D patch on the 3D model.

6 User Interactions

To control and modify the covering process, patches can be modified by placing seeds and selecting seams. These modifications can be done via simple interactions with the system that are described in the following section.

Placing Seeds: If seams should not appear in a specific region on the surface due to their visibility or importance, a seed can be placed in that region. Due to the existence of the seed in the region, the faces in that region are combined into one patch which pushes the seams further. In Cover-it, a seed can be selected using a simple mouse interaction.

Simplifying the Mesh: Many details of the mesh are not visible after covering it with materials. As a result, in Cover-it, the mesh can be simplified in the interest of having smaller number of patches. A mesh is automatically simplified through deleting vertices and its surrounding triangles. The metric is a linear combination of Gaussian curvature estimation and the area of the triangles surrounding a vertex. The holes created after removing the vertex with the smallest Gaussian curvature are filled using a constrained Delaunay triangulation. Fig 10 illustrates that the number of patches can be drastically reduced from 25 to 9 without losing much detail, as opposed to the methods proposed in [4].

Offsetting: Materials that are used to cover a 3D printed objects usually have a thickness. Covering a 3D printed object with a thick material results an inflated covered shape far from the original desired object. In Cover-it, we use an offsetting method [12] to compensate this artifact. Offsetting the surface proportional to the thickness of the material produces a closer shape to the original desired object (see Fig 11). Note that both simplifying the mesh and offsetting are interactive features in Cover-it (please see the supplementary video).

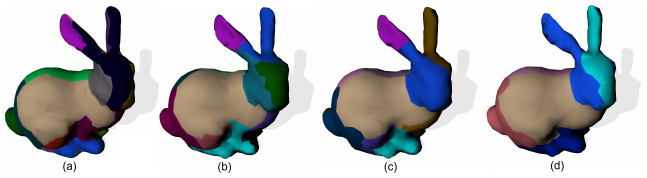

Figure 10: Bunny (a) 25 patches and 3002 faces, (b) 21 patches with 2002 faces, (c) 16 patches and 1602 faces (d) 9 patches and 1202 faces.

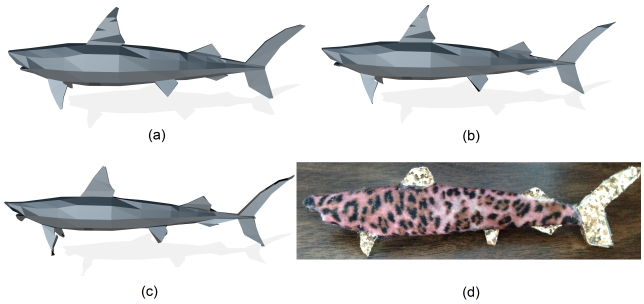

Figure 11: (a) The original model of shark. (b), (c) The offset model of (a) by respectively 0.02, and 0.04 percent. (d) Panther shark created by covering model (c). It is apparent that the thickness of the material and glue has provided an overall shape similar to (a).

Smoothing the Boundary: To control the smoothness in the fitness function, we provide a brush tool to smooth the boundary of each patch. By using this tool, the strength of the smoothness factor is increased if the total fitness function is within a given threshold (see Fig 12). This means that if adding a face from one patch to another does not produce significantly poorer patches (i.e. does not add a face with a very low fitness value), the faces are transferred. Otherwise, faces stay in their original patches.

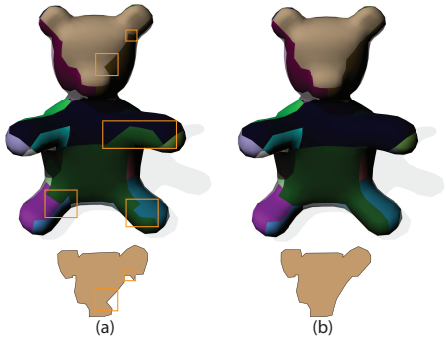

Figure 12: (a) The model before smoothing the boundaries. Boundaries that are intended to be smooth are highlighted. (b) The model after smoothing the boundaries. One of the 2D patches is shown before and after smoothing.

Choosing Seams: The patch boundaries show the cuts that should be applied to materials in order to cover the 3D print. These cuts are seams made up of a set of edges on the mesh. In Cover-it, these seams occur on the boundary between faces that cannot belong to the same patch due to the fitness value. However, to better represent some parts of the mesh, specially the geometric features, it is possible to specify a set of edges to be a seam on the mesh. These seams can be the result of a feature detection algorithm, or

they can be specified by the user (see Fig 13). In Cover-it, a brush tool is provided to select and deselect the seams. After specifying the seams, two faces sharing an edge that belongs to a seam cannot belong to the same patch.

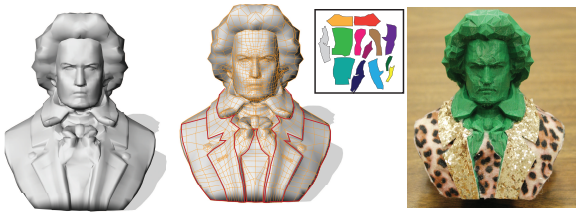

Figure 13: Some feature curves on the 3D model of Beethoven are selected interactively and patches are crafted according to the feature curves. The 3D print of the model is covered by two different materials according to the feature curves.

7 Results and Discussion

In this section, we provide some results of Cover-it. After including pasting guides in the system for wrapping 3D prints, a significant improvement to the results were achieved as the average time that participants took to cover a bunny was reduced from ~ 45 minutes to ~ 25 minutes. These results also featured a reduction in the number and size of gaps and misalignment. Fig 14 shows the results of the pasting process, before and after providing the guidelines (we used a material that shows the seams properly).

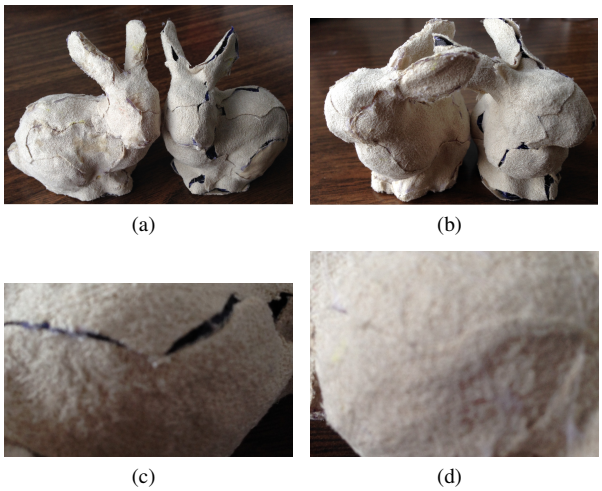

Figure 14: (a),(b) Left: Covered print after providing guidelines. Right: Covered print before providing guidelines. It is apparent that the seams are significantly reduced when guidelines are provided. (c) Seams do not meet without guidelines. (d) The same seams in (c) perfectly meet on the 3D print.

We used Cover-it to make various models with different materials (see Fig 16 and Fig 15). Table 1 indicates, for each model, how many patches are created by Cover-it and how much time (in minutes) we spent to make the final 3D covered model. In Table 1, we only report pasting time. The timing for the cutting phase has been excluded as it can be done manually or using cutter printers. Note that two different 2D layouts were generated for the wolf model; one for each employed material.

In Cover-it, creating the 2D layout is very fast and provides interactive results. This is important since the user can see the result of a set up for parameters and the locations of seeds real time and modify them accordingly. Table 2 lists the time of segmentation for some examined models.

Table 1: Number of faces, and number of patches for each model as well as time required to cover the 3D print.

Model	# of Patches	# of triangles	Time
Beethoven	13	5030	9
Teddy	17	8704	24
Venus	12	708	20
Bunny	14	1250	22
Brown Wolf	18	542	27
Furry Wolf	12	542	10

Table 2: The time (seconds) for segmenting models.

Model	Segmentation Time
Beethoven	0.76
Teddy	0.86
Venus	0.43
Bunny	1.02
Wolf	0.34

8 Conclusion and Future Work

Cover-it uses a multiple seed region growing approach. It also provides simple interactions to improve the patches, such as brush tools to smooth the boundary and choose seams on the patterns. Some gluing guidelines, including color coding, patch boundary carving on the 3D prints, and an animation showing the orientation of each patch are also provided. 2D patches on the layout is also packed using a heuristic method. As available 3D printers are unable to print objects in different materials, we believe that Cover-it can be helpful in designing objects that are required to be using different materials and colors.

Materials used for covering 3D prints may carry different physical properties that should be considered in making the cutting layout. We can use a parameter setting that is pre-defined for common materials such as fur, leather, and paper. To define such settings, we may need to analyze the physical properties of materials.

Figure 15: All the models together.

References

[1] A. Albano and G. Sapuppo. Optimal allocation of two-dimensional irregular shapes using heuristic search methods. *Systems, Man and Cybernetics, IEEE Transactions on*, 10(5):242–248, May 1980.

[2] J. Bazewicz, P. Hawryluk, and R. Walkowiak. Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Annals of Operations Research*, 41(4):313–325, 1993.

[3] J. Blazewicz, M. Drozdowski, B. Soniewicki, and R. Walkowiak. Two-dimensional cutting problem - basic complexity results and algorithms for irregular shapes. *Foundations of Control Engineering*, 14(4):137,160, 1989.

Figure 16: Some models resulting from Cover-it and their 2D layouts.

[4] D. Chen, P. Sitthi-amorn, J. T. Lan, and W. Matusik. Computing and fabricating multiplanar models. *Computer Graphics Forum*, 32(2):305–315, 2013.

[5] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 905–914. ACM, 2004.

[6] Y. Igarashi and T. Igarashi. Pillow: Interactive flattening of a 3D model for plush toy design. In *Smart Graphics*, volume 5166 of *Lecture Notes in Computer Science*, pages 1–7. 2008.

[7] Y. Igarashi, T. Igarashi, and H. Suzuki. Interactive cover design considering physical constraints. *Comput. Graph. Forum*, 28(7):1965–1973, 2009.

[8] D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum*, 24(3), 2005.

[9] M. Kilian, S. Floery, Z. Chen, N. J. Mitra, A. Sheffer, and H. Pottmann. Curved folding. *ACM Transactions on Graphics*, 27(3):1–9, 2008.

[10] S. Laine and T. Karras. Two methods for fast ray-cast ambient occlusion. In *Proceedings of the 21st Eurographics Conference on Rendering*, EGSR'10, pages 1325–1333, 2010.

[11] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3D exploded view diagrams. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 101:1–101:7, 2008.

[12] S. Liu and C. C. Wang. Fast intersection-free offset surface generation from freeform models with triangular meshes. *Automation Science and Engineering, IEEE Transactions on*, 8(2):347–360, April 2011.

[13] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.*, 25(3):681–689, 2006.

[14] K.-Y. Lo, C.-W. Fu, and H. Li. 3D polyomino puzzle. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 157:1–157:8, 2009.

[15] F. Massarwi, C. Gotsman, and G. Elber. Papercraft models using generalized cylinders. In *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*, pages 148–157, Oct 2007.

[16] J. Mitani and H. Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 259–263, 2004.

[17] Y. Mori and T. Igarashi. Plushie: An interactive design system for plush toys. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, 2007.

[18] K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert. Developable surfaces from arbitrary sketched boundaries. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 163–172, 2007.

[19] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pages 146–155, 2003.

[20] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[21] I. Shatz, A. Tal, and G. Leifman. Paper craft models from meshes. *The Visual Computer*, 22(9-11):825–834, 2006.

[22] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov. ABF++: Fast and robust angle based flattening. *ACM Trans. Graph.*, 24(2):311–330, Apr. 2005.

[23] S. Shlafman, A. Tal, and S. Katz. Metamorphosis of polyhedral surfaces using decomposition. *Computer Graphics Forum*, 21(3):219–228, 2002.

[24] M. Skouras, B. Thomaszewski, P. Kaufmann, A. Garg, B. Bickel, E. Grinspun, and M. Gross. Designing inflatable structures. *ACM Trans. Graph.*, 33(4):63:1–63:10, July 2014.

[25] P. Song, C.-W. Fu, and D. Cohen-Or. Recursive interlocking puzzles. *ACM Trans. Graph.*, 31(6):128:1–128:10, 2012.

[26] R. Straub and H. Prautzsch. Creating optimized cut-out sheets for paper models from meshes. Karlsruhe Reports in Informatics 36, Department of Informatics, Karlsruhe Institute of Technology, 2011.

[27] T. Tachi. Origamizing polyhedral surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 16(2):298–311, 2010.

[28] S. Takahashi, H.-Y. Wu, S. H. Saw, C.-C. Lin, and H.-C. Yen. Optimized topological surgery for unfolding 3d meshes. *Computer Graphics Forum*, 30(7):2077–2086, 2011.

[29] C. Wang, Y. Zhang, and H. Sheung. From designing products to fabricating them from planar materials. *Computer Graphics and Applications, IEEE*, 30(6):74–85, Nov 2010.

[30] C. C. Wang. Flattenable mesh surface fitting on boundary curves. *Journal of Computing and Information Science in Engineering*, 8(9-11):1–10, 2008.

[31] S. Xin, C.-F. Lai, C.-W. Fu, T.-T. Wong, Y. He, and D. Cohen-Or. Making burr puzzles from 3D models. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 97:1–97:8, 2011.

[32] H. Yamauchi, S. Gumhold, R. Zayer, and H.-P. Seidel. Mesh segmentation driven by gaussian curvature. *The Visual Computer*, 21(8-10):659–668, 2005.

[33] D.-M. Yan, W. Wang, Y. Liu, and Z. Yang. Variational mesh segmentation via quadric surface fitting. *Comput. Aided Des.*, 44(11):1072–1082, 2012.