Synthesis of Filigrees for Digital Fabrication

Weikai Chen* Xiaolong Zhang* Shiqing Xin† Yang Xia‡ Sylvain Lefebvre§ Wenping Wang*
*The University of Hong Kong †Ningbo University ‡Dalian University of Technology §INRIA

Figure 1: *With the filigree synthesis technique proposed in this paper applied to input surfaces (b) and (f), the generated models (c-e) generally present more fascinating and aesthetic appearance than the base models. Note that the primitive filigree patterns used for (c-e) are shown in (a), whose backgrounds are respectively painted in blue, orange and green. The input models in (b), (d) and (f) are courtesy of Kevin Xu, Open3DModel and PinShape, respectively.*

Abstract

Filigrees are thin patterns found in jewelry, ornaments and lace fabrics. They are often formed of repeated base elements manually composed into larger, delicate patterns. Digital fabrication simplifies the process of turning a virtual model of a filigree into a physical object. However, designing a virtual model of a filigree remains a time consuming and challenging task. The difficulty lies in tightly packing together the base elements while covering a target surface. In addition, the filigree has to be well connected and sufficiently robust to be fabricated. We propose a novel approach automating this task. Our technique covers a target surface with a set of input base elements, forming a filigree strong enough to be fabricated. We exploit two properties of filigrees to make this possible. First, as filigrees form delicate traceries they are well captured by their skeleton. This affords for a simpler definition of operators such as matching and deformation. Second, instead of seeking for a perfect packing of the base elements we relax the problem by allowing appearance preserving partial overlaps. We optimize a filigree by a stochastic search, further improved by a novel boosting algorithm that records and reuses good configurations discovered during the process.

We illustrate our technique on a number of challenging examples reproducing filigrees on large objects, which we manufacture by 3D printing. Our technique affords for several user controls, such as the scale and orientation of the elements.

Keywords: medial axis, Hausdorff distance, filigree synthesis, digital fabrication

Concepts: •**Computer graphics** → **Computational geometry and object modeling;** *Physically based modeling;*

1 Introduction

Filigrees are fascinating ornamental patterns, forming delicate and intricate traceries in space. Their unique aesthetics emerge from the repetition of similar elements arranged in larger patterns, suggesting shapes and volumes. Filigrees often appear as jewels made of thin gold or silver wires, bended and soldered together; but they also appear as laces and finely engraved drawings on glass panels and metal plates. Due to their intricate and delicate nature, fabricating filigrees is an art reserved to few artists mastering highly specialized crafting skills.

The advent of digital fabrication technologies such as 3D printing and laser cutting holds the promise to make these traditional art forms more accessible, and to apply them in contexts that would not be achievable by traditional means. Three such examples are the sculptures *Crania Anatomica Filigree* by Joshua Harker [2011], the magnificent Seashell dress by Fashion House SHIGO [2009], and the concrete filigree enclosing the MuCEM museum [2013].

While digital fabrication simplifies the physical realization of filigrees, a digital model is required before fabrication. In this paper we aim at providing algorithms that can assist the process of modeling filigrees. In particular we focus on the most time consuming task, which is to cover a target surface with a large number of basic elements, while enforcing connectivity and fabrication constraints.

At a high level our approach replicates the traditional filigree design process: The user inputs a set of basic filigree elements that are then automatically repeated, distributed and assembled into a larger pattern covering the target surface. Our algorithm strives to preserve

SIGGRAPH '16 Technical Paper, July 24 - 28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925911

the appearance of each individual element, joining them in natural ways. The produced filigrees are fully connected and optimized to minimize fragilities, and are ready for 3D printing. In addition our technique provides many user controls: multiple base elements can be specified; orientation and scale fields can be defined along the target surface.

At a technical level, we solve for an element packing problem along the target surface. Tightly packing elements of arbitrary shapes along surfaces is extremely challenging. To achieve this task we exploit two specific properties of our problem. First, filigrees are well described by their skeletons, allowing for simple manipulations such as detecting overlaps, performing local deformations or pruning spurious branches. Second, the repetitive nature of filigrees affords for additional degrees of freedom. Elements can be partially overlapped in inconspicuous ways, relaxing the packing problem. We formalize the quality of pattern overlaps through a novel partial Pattern Matching Energy (PME) based on the modified Hausdorff distance. We exploit this energy in a stochastic optimization scheme. Starting from a dense packing of many elements — guaranteeing full connectivity but having many overlaps — we progressively refine the result. Refinements are performed through local optimization of element positions, accepting overlaps having a low matching energy. Through a limited amount of deformation our technique preserves the global connectivity of the pattern while resolving overlaps. A key component of our technique is a novel approach for recording and reusing good configuration between pairs of elements. We call this approach *boosting* as it progressively encourages good matches and good overlaps to appear. Structural properties are jointly optimized by encouraging additional connections to appear in fragile areas.

Our contributions are:

- The formulation of filigree synthesis as a packing problem with appearance driven partial matches, based on a novel pattern partial matching energy.
- A stochastic optimizer that produces fully connected patterns with good structural properties after fabrication.
- A novel boosting strategy to record and reuse good matches discovered during the stochastic exploration.

2 Previous Work

Texture synthesis The goal of by-example texture synthesis is to reproduce a colored pattern resembling a small exemplar given as input. This is typically done for texture images, generating larger extents of pattern while avoiding obvious periodicities [Wei et al. 2009]. Several algorithms are able to synthesize a texture from an example along a surface, e.g. producing per-vertex colors along a dense mesh [Turk 2001; Wei and Levoy 2001], directly in texture maps [Zhang et al. 2003; Lefebvre and Hoppe 2006], or by covering the model with patches [Praun et al. 2000]. Mesh Quilting [Zhou et al. 2006] extend these techniques to geometry: the input texture is a patch of geometric *texture* that is used to cover a target surface. It is expected that the left/right top/bottom boundaries contain similar, repeating content. The surface is covered by placing each patch in sequence, cutting its content to best match previously placed patches. The approach is not designed to work for individual elements, and without repeating features it cannot find good cuts to resolve seams. It therefore does not apply to the context of our approach. Zhou et al. [2014] consider the synthesis of patterns along curves while constraining the topology of the result. This allows the fabrication of singly connected ornaments along curves, but does not extend to higher dimensions (2D/3D).

Closer to our purpose, the work of Dumas et al. [2015] synthesizes a fabricable pattern along a surface. The input is a stochastic pattern defining solid/empty regions. The approach takes into account structural properties and fabrication constraints. Martinez et al. [2015] investigate automatic shape design under rigidity and appearance objectives. The appearance is defined by an input exemplar pattern. While these techniques excel at producing stochastic, organic patterns, they cannot properly capture relatively large individual elements. This stems from the Markov Random Field assumption that limits appearance to a local definition. Li et al. [2011] use field-guided shape grammars to synthesize geometric patterns. The grammar rules are manually designed from input patterns. However, this approach would generate artifacts when patterns are densely placed, limiting its application for digital manufacturing.

Several approaches have been proposed that focus specifically on the synthesis of element arrangements, e.g. [Ma et al. 2011; Hurtut et al. 2009]. The input captures both a set of disjoint elements and their spatial relationships. A similar distribution of non-overlapping elements is synthesized. In contrast our work inputs only a set of *independent* elements — there is no target spatial arrangement specified in the input for our algorithm to mimic — and our technique exploits potentially large overlaps between elements for generating fully connected filigree patterns. In addition, as we target fabrication the connectivity between patterns is crucial to ensure the rigidity of the final printouts.

In concurrent work Zehnder et al. [2016] synthesize filigrees by packing curved elements along surfaces. The approach provides an interactive authoring tool that can automatically generate an initial packing. The curves elastically deform, and their positions are optimized to reduce deformations while enforcing contact and sizing constraints. The system reveals weak areas that the user can reinforce by interactive editing. The elements are not allowed to overlap and may be large compared to the surface curvature. In contrast we focus on fully automatic synthesis with large numbers of curve elements, exploiting overlaps and reinforcing the structure automatically. Both approaches are complementary and the deformation analysis and optimization in Zehnder et al. [2016] would benefit our work.

Structural analysis for fabrication Our technique considers the structural properties of the final object. Several techniques have been proposed to help user identify and fix weaknesses of an input object. Stava et. al [2012] automatically add struts to an object after identifying weakness by the finite element method. Zhou et. al [2013] perform a worst case analysis to identify weak regions of a 3D print without prior-knowledge of external forces. Umetani and Schmidt [2013] perform a fast, interactive cross sectional analysis to present the user with a weakness map. Our work performs a structural analysis that is specifically tailored to our needs, simulating the external surface with shell elements. Instead of adding visible struts, we locally change the thickness of the filigree and add more elements to locally reinforce the filigree.

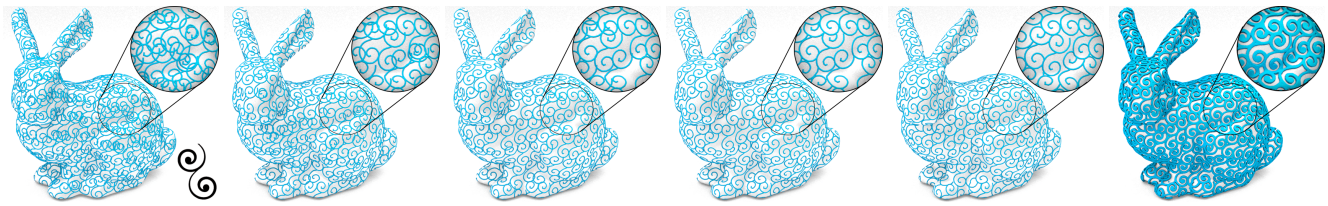
3 Filigree Synthesis

3.1 Terminology

First we fix the terminology. A *filigree pattern* refers to a design layout of curvilinear strips, often called *traceries*. A basic filigree pattern used to compose a large pattern is called a *filigree pattern element*, or *element* for short. An element often again consists of individual curvilinear strips, which will be called *branches*.

3.2 Input and Output

The input to our synthesis algorithm consists of some 2D exemplar filigree elements and a base surface serving as the output domain



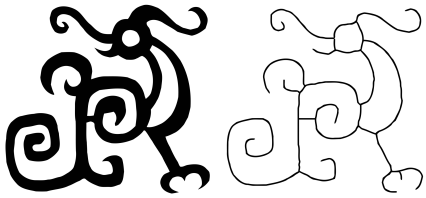
(a) *Initialization* **(b)** *Placement Optimization* **(c)** *Boosting* **(d)** *Topology Cut* **(e)** *Structural Optimization* **(f)** *Reconstruction*

Figure 2: *Filigree synthesis pipeline. We first (a) generate an initial element distribution over the base model (the input element is shown on the right bottom); (b) the placement of element is first optimized using stochastic search with connectivity constraints; then followed by (c) a boosting step to improve overall element distribution. (d) Topology cut is applied to trim off conflicting branches. (e) Structural optimization strengthens the weak regions via adding new elements. (f) A final model is reconstructed that is ready for digital fabrication.*

for filigree synthesis. The output is a visually pleasing filigree layout over the base surface which is composed of well-connected and partially overlapping duplicates of the exemplar element, thus making the synthesized pattern have a similar style to the input exemplar elements. The base surface may be equipped with a user-specified control field which dictates how the composing elements should be oriented and scaled over the base surface.

3.3 Medial Axis Representation

A typical filigree pattern consists of connected curvilinear strips, as shown in Figure 3a. For any filigree pattern, because of the curvilinear nature of its constituent strips, we will use the medial axis of the pattern to represent its main skeletal structure, and call its medial axis the *skeleton graph* of the pattern. To simplify geometric processing tasks during filigree synthesis, we will mainly work with the skeleton graphs of filigree patterns and approximate the skeleton graphs by polygonal curves with user-specified accuracy, as shown in Figure 3b. Towards the end of our algorithm, we will obtain a large skeleton graph that can be viewed as the medial axis of the synthesized filigree design. By recovering width to convert this large skeleton graph into connected strips, we yield the final synthesized filigree design on the base surface (see Figure 1d).

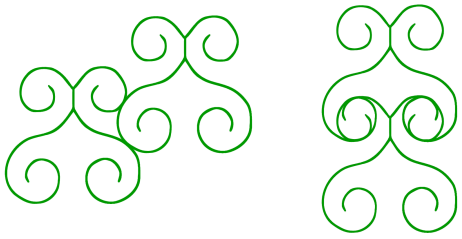


(a) **(b)**

Figure 3: *(a) A typical filigree pattern consists of connected curvilinear strips. (b) The skeleton graph of the filigree element in (a).*

3.4 Requirements

It is required that the synthesized filigree elements need to be connected into a single piece with sufficient mechanical strength. We encourage two types of connections to ensure preservation of the style of the exemplar filigree element: (1) *tangential connection*, which means smooth and natural contacts between the branches of adjacent elements (see Figure 4a); and (2) *partial overlapping with matching shapes*, in which case two adjacent elements overlap partially, and the geometrical shapes of the two elements match well within the overlapping region (see Figure 4b). The quantitative measurement of shape matching will be elaborated later.



(a) **(b)**

Figure 4: *(a) Tangential connection. (b) Partial overlap.*

3.5 Pipeline Overview

The pipeline of our synthesis algorithm has the following main steps, as shown in the flowchart in Figure 2.

Step 1: Initialization In the initialization step, the duplicates of the exemplar filigree element are distributed over the base surface using a blue-noise sampling method. This initial distribution takes into consideration the control field as well as the unbiased selection of input elements in the case where multiple basic filigree elements are provided. The resulting distribution of the filigree elements at this stage ensure sufficient coverage of the base surface and close connection between the elements, but are visually unsatisfactory since the pattern elements overlap in a random manner.

Step 2: Placement optimization In this step we locally adjust the positions and shapes of all the filigree elements to achieve visually more pleasing connections between elements, while preserving the surface coverage and inter-element connectivity. Two kinds of operations are performed in this stage to improve the visual quality of the connections between adjacent elements: (a) *Position adjustment*: Two elements that overlap partially will have their relative positions adjusted by small translational displacement to achieve a better matching of their shapes within the overlapping region. (b) *Forcing tangency*: If two elements are nearly in tangential contact between their curvilinear branches, then the elements are brought into tangent contact via non-rigid deformation.

Step 3: Boosting The local search method employed in the preceding step is easy to get stuck in a local minima. A simple but effective modification is to iteratively call the local search routine, but starting from a different initial configuration at each time. We, therefore, propose a boosting step to improve the overall pattern distribution via learning good relative positions between element pairs. The candidate pairs are progressively recorded in a database when numerous position adjustments are attempted in Step 2 on placement optimization. Any pair of neighboring elements that cannot be put in a good connection will be replaced by a better-connected pair that is learnt from the database. The output element distribution

will be fed back to the second step in an iterative manner.

Step 4: Topology cut While satisfactory connections can be achieved for most pairs of adjacent elements, some pairs of adjacent elements may still have unacceptable connections with each other despite the efforts on improving partial overlap in step 2 and 3, as shown as Figure 2d. This is typically manifested by some conflicting branches of the elements that intersect each other, causing undesired visual artifacts. To fix this, we apply an *topology cut* operation to trim off the conflicting branches with lower importance.

Step 5: Structure optimization The skeleton graph obtained after topology cut is first converted into surface mesh representation by adding the width to the curve segments of the graph. The filigree design obtained is connected and visually satisfactory, but may lack needed mechanical strength to endure normal or specified handling. We apply structural optimization to iteratively improve pattern coverage and enhance element connections until sufficient model rigidity is reached. Finally, the improved skeleton graph is converted to the 3D mesh representation that is ready for fabrication.

Discussion We wish to emphasize that we allow complete overlap between elements in the second and third step if our solver find such movement is helpful to improve the matching quality. We only keep one copy of the entirely overlapped elements when the iterations stop at each step. This strategy helps us to control the element number in an implicit way. Our method will end up with a proper number of elements regardless an initialization with excessive elements. Note that in the steps before structure optimization, all the filigree elements are represented merely by their skeleton graph.

4 Pattern Synthesis

In this section we will present the details of the core algorithms of filigree synthesis, that is the first four steps in the pipeline: (1) initialization (2) placement optimization (3) boosting and (4) topology cut. As these four steps mainly determine the appearance of final output, we call them pattern synthesis in the rest context of this paper.

4.1 Initialization

The goal of initialization is to distribute the instances of the exemplar elements such that there is sufficient coverage of the base surface. Furthermore, it is required that each element is well connected with its neighboring elements. Our initialization method follows the dart-throwing approach in blue-noise sampling problem [Wei 2010]. A fixed point of each exemplar filigree element is picked as its reference point. We repeatedly generate sample points on the base surface by dart throwing around the boundary of the existing elements. For each sample point generated, the instance is kept only if it overlaps with some existing elements and the overlapping area satisfies certain conditions to be elaborated below; otherwise it is rejected and another sample point is generated. This is repeated until the entire base surface is sufficiently covered and no more element can be added.

Specifically, let S_i denote the current instance of the element that newly generated. Then S_i is accepted if and only if all the following conditions are met: (1) The overlapping area of S_i with the union of all the existing elements is no more than 60% of the area of S_i; (2) The overlapping area of S_i with any of the existing elements is no more than 30% of the area of S_i; and (3) The overlapping area of S_i with at least one of the existing instances is no less than 10% of the area of S_i. These conditions ensure that the newly accepted element covers some previously uncovered regions of the base surface, while having sufficient overlap with the existing elements to ensure close connection between adjacent pieces. If the base surface has a control field, each generated element should be rotated and scaled as dictated by the control field before testing its overlap with the existing elements.

To ensure unbiased distribution of the exemplar filigree elements in the case that multiple exemplars are provided as input, we follow the strategy in [Wei 2010] for multi-class blue noise sampling to pick the next trial pattern from the class that is currently most underfilled. Furthermore, a pattern element of a larger size is given a lower priority to be picked. Here, the size of an element pattern is defined to the diagonal length of the bounding box of the pattern.

4.2 Pattern Matching Energy

The randomly distributed pattern elements generated by initialization often have highly undesirable visual artifacts, although they are connected and cover the base surface well, as shown in Figure 2a. These artifacts are mainly due to that most adjacent patterns overlap in a random manner, without having their geometric features aligned with each other. We now propose some measures on the visual quality of the connection between adjacent elements.

First we discuss how to quantitatively measure the quality of alignment of two partially overlapping pattern elements. For a filigree pattern element, we need to define its *covering region*. If the element is convex or nearly convex, we simply take the convex hull of the pattern to be its covering region. Otherwise, we decompose the pattern element into several convex or nearly convex components and take the union of the convex hulls of these components to be the covering region of the original element (see the left figure as an example, the covering region is marked in blue). We developed a user interface to allow the user to easily perform the convex decomposition of a non-convex element.

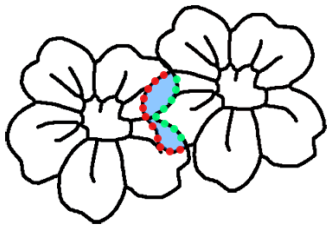


Figure 5: *Overlapping region (shown in blue) between two neighboring elements. The vertices of the polygons of the two elements (shown in red and green) are the input point sets to compute the modified Hausdorff distance.*

For two partially overlapping pattern elements, we define the intersection of their covering regions to be their *overlapping region* (see Figure 5). Then we measure alignment quality of these two elements by the Hausdorff distance between the subparts of the two pattern elements *within* their overlapping region. Since each filigree element is represented by its skeleton graph with edges being polygonal curves, the evaluation of this Hausdorff distance is reduced to the computation of the Hausdorff distance between two finite sets of points which are the vertices of the polygons from the two skeleton graphs that lie in the overlapping region (see Figure 5).

We adopt the *modified Hausdorff distance* [Dubuisson and Jain 1994], which has proven to be more effective for shape matching purpose than the standard Hausdorff distance. For two finite point sets U and V, their modified Hausdorff distance is defined to be

$$\mathrm{dist}_{MH}(U,V) = \max\left(\frac{1}{N_U}\sum_{u\in U}\mathrm{dist}(u,V), \frac{1}{N_V}\sum_{v\in V}\mathrm{dist}(v,U)\right) \quad (1)$$

where N_U and N_V are the number of points of U and V, respectively. $\mathrm{dist}(u, V)$ is the closest distance from the point u to the point set V, i.e. $\mathrm{dist}(u, V) = \min_{v \in V} d(u, v)$, where $d(u, v)$ is the Euclidean distance between the points u and v. The term $\mathrm{dist}(v, U)$ is similarly defined.

For two overlapping elements P_i and P_j, let U_i' and V_i' denote the set of the vertices of the skeleton graphs of P_i and P_j, respectively, that lie in the overlapping region of P_i and P_j. Then the matching quality of the patterns P_i and P_j is defined to be

$$\mathrm{dist}(P_i, P_j) = \mathrm{dist}_{MH}(U_i', V_i') \tag{2}$$

Intuitively, this term measures how well the shapes of the two filigree elements P_i and P_j match each other within their overlapping region. Note that if P_i and P_j do not overlap, that is U_i' and V_i' are both empty set, we penalize $\mathrm{dist}(P_i, P_j)$ to be infinity. This setting helps to enforce pattern connections during minimizing the global matching energy that defined below.

Now we are ready to define a global energy function, called *Pattern Matching Energy* (PME), to measure the overall quality of the synthesized filigree pattern. Let P_i denote a pattern element. Let Γ be the set of the index pairs (i, j) such that the elements P_i and P_j are connected to each other. Γ can also be viewed as the edge set of the connectivity graph of all the pattern elements. Let $\mathcal{P} = \{P_i\}$ denote the set of existing patterns. Then the PME function is defined to be

$$E(\mathcal{P}, \mathcal{O}) = \sum_{(i,j) \in \Gamma} \mathrm{dist}(P_i, P_j) + \Theta(\mathcal{P}, \mathcal{O}) \tag{3}$$

where the first term measures the overall appearance by considering the alignment quality of every pair of connected pattern elements. Θ is an optional application-specific energy term, which can be cast into connectivity constraints among $\mathcal{P}$ or field constraint to follow the requirements of control field over the output domain $\mathcal{O}$. Our goal is to find an element distribution $\mathcal{P}$ with lowest PME value.

4.3 Placement Optimization

The goal of this step is to improve the initial distribution of pattern element $\mathcal{P}$ by minimizing the PME energy. However, it is non-trivial to perform a meaningful gradient descent to minimize PME in Equation 3 due to the non-linearity of Hausdorff distance, not mentioning that arbitrary pattern shapes could be taken as input. Therefore, we resort on a greedy strategy to iteratively refine the placement of each element. In order to maintain connectivity between pattern elements, in this step, we impose hard constraints to maintain the overlapping relationship between element pairs. That is, the initial neighbors $\mathcal{N}_i$ of element P_i should at least be the subset of its new neighbors $\mathcal{N}_i'$ after placement optimization.

There are two phases of placement optimization as shown in Algorithm 1.

4.3.1 Stochastic Search via Translation

Stochastic search is implemented in StochasticSearch. In this phase, each element undergoes some small translational displacement around its current position to search for a location with a lower value of the PME energy function, indicating better alignment of the element with its neighboring elements. These displacements are generated by randomly sampling a number of points as the candidate positions of the element in the neighborhood of its current location. All these sampling points are tested and the position with the lowest PME value is returned as the optimal position in current iteration. Note that for each candidate position, we need to rotate and scale the element according to the underlying control field before computing the pattern matching energy.

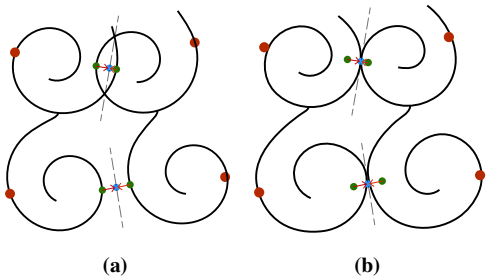


Figure 6: *(a) A double intersection and a small gap between neighboring elements are forced to be in tangent contact by deformation in our algorithm. The moving control points are shown in green while the red ones are fixed control points. (b) Tangent contacts are achieved after non-rigid deformation.*

In order to satisfy the hard connectivity constraints, we first find the neighbors $\mathcal{N}_i$ of P_i in GetNeighbors before updating its position. During stochastic search, we observe that some new neighbors $\mathcal{N}_i'$ of P_i may result from the candidate positions. Such new neighboring relation is accepted, as long as it yields the smallest PME value. Note that during stochastic search, we record data (i.e. relative position, class ID *etc*) of all element pairs that have been traversed and output as connection database $\mathcal{D}$ for the learning purpose in the following boosting step.

Algorithm 1 PlaceOptim

Input:
Input surface model $\mathcal{S}$; Initial placement of pattern set $\mathcal{P}$ on S; Input control field $\mathcal{F}$ on $\mathcal{S}$

Output:
An optimized pattern placement $\mathcal{P}_O$ following $\mathcal{F}$ on S that conforms to both input connectivity constraints and control field $\mathcal{F}$; A connection database $\mathcal{D}$ that records all element pair with their matching quality;

```
Π ← BuildConnectivityGraph(P);
{Ω_i} ← IndependentSet(Π);
while true do
    for each Ω_i ∈ {Ω_i} do
        for each P_j ∈ Ω_i do
            N_i ← GetNeighbors(P_i, Π);
            StochasticSearch(P_i, N_i);
            accept ← SmoothConnection(P_i);
            if accept then
                update the shapes of P_i and its neighboring elements;
            end if
        end for
    end for
    P ← Update();
    if converged or enough # of iterations reached then
        break;
    end if
end while
return P_O = P;
```

4.3.2 Smooth Connection via Non-rigid Deformation

During placement optimization, although each pair of overlapping elements are aligned better than before, we observe that their geometric features could often match better to be visually more pleasing if the elements can be modified by a non-rigid deformation of limited amount.

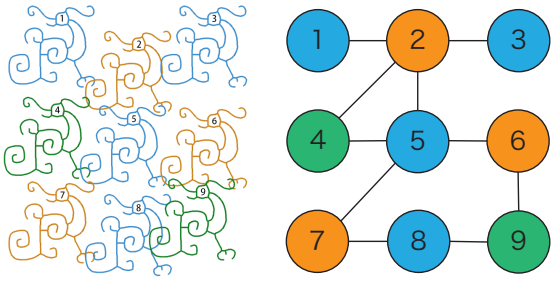
We encourage tangent connection throughout our synthesis pipeline as it is one of most visually pleasing manners to join two neigh-

boring elements. There are two cases, as shown in Figure 6, that deserve special treatment. That is, (1) the two curves have two intersection points that are close to each other; and (2) the two curves segments are separated by a narrow gap. In both cases we use non-rigid deformation to bring the two curves into tangential contact with each other. In case (1), the tangential contact keeps the connection between the two elements but make them contact in a smoother way, thus improve the visual quality of the synthesized pattern. In case (2), the forced tangential contact eliminates the narrow gap, thus again improves the visual appearance of the pattern design, and introducing a new pair of connected elements, if that didn't exist before.

The details of deformation method are elaborated in Section 8.2. Basically, we achieve non-rigid deformation via moving/fixing a set of control points. Take Figure 6a for instance, we first connect the intersection points with a line. The moving control point (shown in green) is picked on the skeleton graph that has the closet tangent with the line slope. The fixed control points (shown in red) are those intersections points with other neighboring elements that do not need to be changed. We then apply the deformation to achieve tangential contacts by matching the moving control points to their projections on the connecting line while letting the fixed control points remain still. See Figure 6b.

Distortion Measurement. As non-rigid deformation may introduce great distortion, we define a cost function to measure the degree of distortion. Given a pattern element P that is discretized into N dense sample points $\{p_i \in P, 1 \leq i \leq N\}$. We use a distance matrix $D_P = (d_{ij})_{N \times N}$ to encode the shape of P, where d_{ij} is the distance between p_i and p_j. Obviously, D_P is independent of translation and rotation. Suppose P is transformed into P' and we can compute distance matrix $D_{P'} = (d'_{ij})$ in a similar fashion. Then we measure the distortion of P' with regard to P using the standard deviation of $\{\frac{d'_{ij}}{d_{ij}}, i \neq j\}$ as the cost function, which gives the measurement on the acceptance/rejection of the deformation.

This phase is implemented in SMOOTHCONNECTION (Algorithm 1) which performs the deformation and returns the flag of accepting the such deformation based on distortion measurement. If the cost of deformation is lower than a tolerance threshold, we update the shapes of P_i and its neighboring elements accordingly.

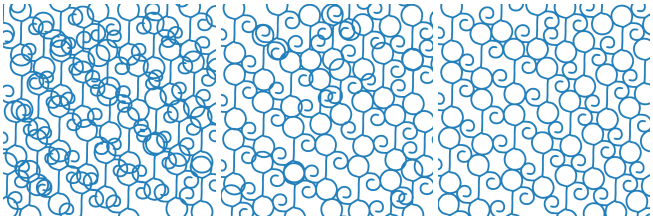


(a) **(b)**

Figure 7: *Pattern distribution (left) and its corresponding node graph (right).*

Updating Sequence If all elements are updated simultaneously, one problem is that elements updated according to neighbors are also changing. We use an updating sequence based on independent set. First, all the elements are divided into several independents sets such that any two elements in the same set are not connected (as illustrated in Figure 7b). Then the elements in each set are updated before processing those in the next set. This idea is similar to the *subpass* strategy in [Lefebvre and Hoppe 2005], which greatly improves their correction performance. Our experiments showed that the convergence of this strategy is much better than using a depth-first or breadth-first processing order in the connectivity graph of the elements. Indeed, one may also adopt a random order to traverse all the elements to optimize their positions; however, our scheme has the potential advantage of allowing parallel processing in this time-consuming task of optimizing all the elements in multiple rounds, since it is ensured that any two elements in the same set are always not connected so can be updated simultaneously without producing conflicting updated positions. In Algorithm 1, BUILDCONNECTIVITYGRAPH computes the connectivity graph among input elements. INDEPENDENTSET then generates the independent sets afterwards.

Figure 8 shows how the initial layout of the pattern elements is improved after the step of placement optimization.



(a) *Initialization* **(b)** *After placement optimization* **(c)** *After boosting*

Figure 8: *Illustration of the effect of each step.*

4.4 Boosting

Because the minimization of the PME function is a difficult non-convex and combinatorial optimization problem, the resulting pattern layout after placement optimization may still leave some pairs of connected elements with unsatisfactory alignment. Specifically, while the hard constraint in the preceding step is helpful to maintaining the connectivity of the synthesized filigree pattern, it makes our search prone to getting trapped in a poor local minimum, thus hampering the ability of finding a better pattern layout with an even smaller PME energy value. For example, the element placement shown in Figure 8b cannot be further optimized by placement optimization.

We remedy this issue by using a boosting strategy that replaces a pair of elements with unsatisfactory alignment by the same pair with better alignment, while relaxing the constraint that the connectivity of each involved element with its neighboring element be the same as before. Hence, this operation enables us to search for a better pattern layout starting from a new local initialization. In other words, the boosting step aims to improve pattern distribution by providing better local pattern connectivity so that subsequent application of placement optimization can jump out of a poor local minimum. The boosting step is further followed by another round of placement optimization. These steps are iterated until convergence or a satisfactory result is obtained.

The pseudo-code of boosting is illustrated in Algorithm 2. Specifically, there are two phases of implementing this boosting strategy.

Selection Phase During the massive search in preceding step of placement optimization, nearly all pairs of adjacent elements with good alignment have been collected and kept in the database $\mathcal{D}$. The goal of selection phase is to select candidate boosting vectors that for the learning in the next phase. Before selection phase, we filter out those element pairs with low matching quality in FILTERCANDIDATEPAIRS (Algorithm 2). Specifically, we only kept those pairs with best matching quality (typically top 5% out of all the items in our implementation) and obtained a filtered database $\mathcal{D}'$. While updating P_i, we collect boosting vectors from all its neigh-

Algorithm 2 BOOSTING

Input:
Input surface model $\mathcal{S}$; Input pattern set $\mathcal{P}$ on S; Input control field $\mathcal{F}$ on $\mathcal{S}$; Connection Database $\mathcal{D}$;

Output:
An output pattern placement $\mathcal{P}_O$ that resembles the good spacing example in $\mathcal{D}$ while maintaining sufficient connectivity;

1: $\mathcal{D}' \leftarrow$ FILTERCANDIDATEPAIRS($\mathcal{D}$);
2: **while** TRUE **do**
3: $\Pi \leftarrow$ BUILDPATTERNCONNECTIONMAP($\mathcal{P}$);
4: $\{\Omega_i\} \leftarrow$ INDEPENDENTSET(Π);
5: *// Selection Phase*
6: **for each** pattern $P_i \in \Omega_i$ **do**
7: candidate set $C_i := \emptyset$;
8: $\mathcal{N}_i \leftarrow$ find neighbors of P_i;
9: **for each** neighbor $P_j \in \mathcal{N}_i$ **do**
10: $b_j \leftarrow$ FINDBOOSTINGVECTOR(P_i, P_j, $\mathcal{D}'$);
11: $C_i := C_i \cup b_j$;
12: **end for**;
13: $C_i = C_i \cup$ AVERAGE(C_i);
14: *// Learning Phase*
15: Attempt all boosting vector $b_j \in C_i$ and assign P_i with the one with lowest PME value.
16: **end for**
17: **if** converged **or** enough # of iterations reached **then**
18: **return** $\mathcal{P}_O = \mathcal{P}$;
19: **end if**
20: **end while**

bors $\mathcal{N}_i$ in the candidate set C_i. We denote the relative position between (P_i, P_j) as $\nu_{ij} = p_j - p_i$, where p_i and p_j are the local coordinates of P_i and P_j within a locally parameterized surface domain. Similarly, for a candidate pair $(\bar{P}_i, \bar{P}_j) \in \mathcal{D}'$, we can define its relative position vector $\bar{\nu}_{ij}$. Then the similarity metric between (P_i, P_j) and $(\bar{P}_i, \bar{P}_j)$ is formulated as $|\nu_{ij} - \bar{\nu_{ij}}|^2$. For each $P_j \in \mathcal{N}_i$, we query $\mathcal{D}'$ with the element pair (P_i, P_j) to find its best matching pair $(\bar{P}_i, \bar{P}_j)$ with the smallest similarity metric value (i.e. the most similar one). The relative position $\bar{\nu_{ij}}$ of the best match pair $(\bar{P}_i, \bar{P}_j)$ is returned as the boosting vector b_j corresponding to P_j. Note that P_i and $\bar{P}_i$ are the same element (so are P_j and $\bar{P}_j$), but the pairs (P_i, P_j) and $(\bar{P}_i, \bar{P}_j)$ may have different relative vectors. We also append the average vector of all the candidate boosting vectors in C_i as an additional boosting vector (Line 11 in Algorithm 2).

Learning Phase The second phase is to learn from candidate boosting vectors. For each learning attempt, we translate each element with the boosting vector. Note that, since the displacement by the boosting vector does not have to preserve the previous connectivity, there may be significant loss of connection between patterns. When that occurs, we will enhance the connectivity by forcing some nearby pairs of elements to be connected to each other using non-rigid deformation (using identical method in Section 4.3.2). We compute the PME value resulting from each boosting vector and keep the one with highest score.

Stopping Criteria We stop the iteration of *Placement Optimization* and *Boosting* if there is no significant change in the synthesized layout. Specifically, we compute the sum of element translation in each iteration and terminate the iterations if it is below a threshold.

4.5 Topology Cut

After the steps of placement optimization and boosting, there may still exist misalignment between some connected elements and such misalignment are mostly manifested as the intersection of conflicting branches from the two involved patterns. To resolve this issue, we develop an effective scheme, called *topology cut*, for trimming off some conflicting branches within the overlapping region of such two pattern elements. Figure 9 shows how this kind of misalignment is resolved by topology cut to improve the visual quality of the output.

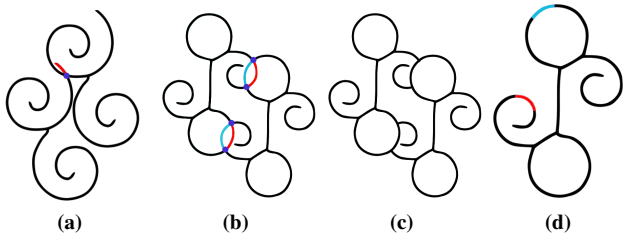


Figure 9: *(a) Short protruding segment (in red) will be trimmed off. (b) Overlapping of similar features will lead to conflicting branches (blue and red). (c) Topology cut result of (b). (d) The blue subgraph has higher importance than the red one as it is inside a loop.*

We consider two cases in our topology cut scheme: (1) trimming off a short protruding curve segment when two curves intersect at one single intersection; and (2) trimming off some branches when there are more than one intersection points that are close to each other. In case (1), as shown in Figure 9a, we simply detect the protruding curve segment that is shorter than some threshold (shown in red) and trim it off.

However, a more elaborate treatment is needed to deal with case (2), where the branches for the two pattern elements intersecting each other in a more complicated manner. Consider two connected patterns, denoted P and Q, whose skeleton graphs intersect at a number of points, t_i, $i = 1, 2, \ldots, m$ (purple dots in Figure 9b). Using the intersection points t_i as cutting points, we decompose each skeleton graph into a number of subgraphs. Then the conflicting branches from P and Q are two of these subgraphs. For example, each pair of blue and red branches in Figure 9b are two curve segments sharing the same endpoints. We shall next assign an importance score to each of these two subgraphs and remove the subgraph with lower importance score.

The importance score of a subgraph in this context is defined as follows. Suppose that the skeleton graph $G(P)$ of the pattern P is decomposed by the intersection points t_i into a collection of connected subgraphs G_k, $j = 1, 2, \ldots, n$. Intuitively, a subgraph G_k receives a lower importance score, i.e., is less important, if (i) G_k has a small size, measured in the total length of all its edges, denoted as $L(G_k)$; or (ii) G_k is located near "frontier" of its supergraph $G(P)$. (The "frontier" of a skeleton graph is understood to be the set of all the valence-1 vertices.) Regarding the latter consideration, we observe that, usually, G_k is near the "frontier" of $G(P)$ if one of the subgraphs of $G(P)$, excluding G_k itself, has a small size. Hence, we measure the proximity of G_k to the "frontier" of $G(P)$ by $F(G_k) = \min_{j \neq k}\{L(G_j)\}$.

Finally, summarizing the two considerations above, we define the importance score of the subgraph G_K as

$$I(G_k) = \alpha L(G_k) + \beta F(G_k), \tag{4}$$

where α and β are weighting coefficients. We use $\alpha = \beta = 1$ in our implementation. Figure 9d provides an example to compare the importance scores of two subgraphs of a skeleton graph. Hence, given two conflicting branches that are represented by two subgraphs from the two connected pattern P and Q, we first evaluate the importance scores of the two subgraphs by Equation (4), with respect to their own supergraphs $G(P)$ and $G(Q)$, respectively. Then we keep the subgraph with the higher importance score,

and trim off the other.

5 Structural Optimization

During structural optimization, we iteratively detect weak regions and reinforce these areas. This iterates until the shape is strong enough. The user can manually specify the initial thickness of model and external force profiles. The external forces are pressure forces from the outside, applied on all shell elements. The force profiles could be easily changed to match different scenarios.

Our structural optimization contains two phases. In the first phase, we apply structural analysis to detect mechanically weak regions of the reconstructed surface mesh (see Section 6 for more details on mesh reconstruction) based on the synthesized filigree design. New pattern elements are then inserted into the weak regions to create denser coverage and connections. We then re-synthesize the result throughout previous stages based on the new pattern layout. We define the weak node as the one with Von Mises stress larger than yield strength σ_{yield}. The first phase is repeated until the portion of weak nodes is below a threshold (typically 0.5% in our implementation). See Figure 10 for an example showing how the mechanical strength of a synthesized filigree pattern is enhanced after the first phase.

After strengthening local weak parts, we obtain a balanced structure with respect to both gravity and external forces. We then iteratively increase the thickness of model in the second phase, by 10% in each iteration, until no weak nodes are detected.

Our structural optimization tries to strengthen the weak parts first before increasing the thickness of model. This pipeline is based on our observations in experiments that if we increase the model thickness first, the weak regions cannot be totally removed even when the thickness is increased significantly. However, we observe that if we strengthen weak regions first to obtain a uniformly balanced structure, the strength of the model can be greatly improved with a small increase in thickness, which leads to faster convergence.

Finite Element Simulation In order to accelerate structural simulation, we apply numerical simulation on open surface mesh with shell element analysis. This treatment leads to faster simulation for two reasons: (1) The generation of volume mesh with specified thickness is time-consuming, while the thin shell model based on a 2D surface mesh can be generated much faster. In fact, a surface mesh with a preset thickness is a good approximation to a closed mesh for the purpose of structural simulation. (2) The computation for analyzing shell elements is much faster than conventional methods based on volume elements [Zienkiewicz and Taylor 2005].

We simulate the elastic material with the properties of commonly used printing materials, typically ABS or PLA plastic. We use Von Mises for stress simulation which is formulated as

$$\bar{\sigma} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1-\sigma_2)^2+(\sigma_2-\sigma_3)^2+(\sigma_3-\sigma_1)^2} \qquad (5)$$

where $\sigma_i (i = 1, 2, 3)$ represents the eigenvalue of the stress tensor. The stress tensor field is calculated from the displacements and rotations at all nodes of input model, following a standard procedure of finite element analysis [Bathe 2006].

Figure 10 shows the intermediate results of structural optimization. As seen from the results, the weak regions (shown in red) are greatly reduced after structural optimization. We also present a complete comparison in Section 8.3.1 that demonstrates the mechanical properties of all tested models before and after structural optimization.

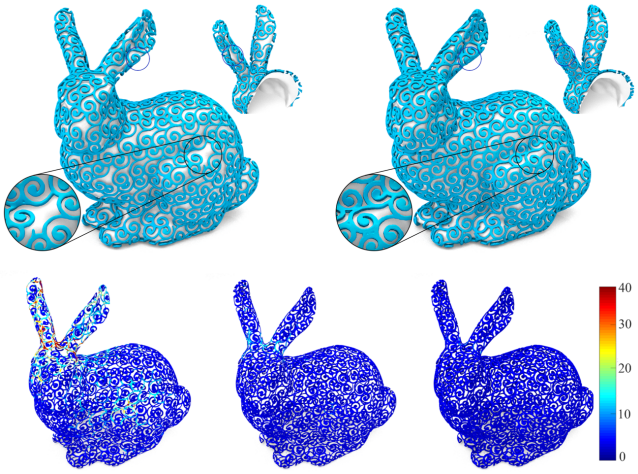


Figure 10: ***Top:*** *From left to right, synthesis result before and after adding more element connections to strengthen weak parts.* ***Bottom:*** *Color coded stresses results during structural optimization. Left: Before structural optimization. Middle: After strengthening local weak regions. Right: After increasing model thickness. The bunny model is courtesy of of Stanford 3D scanning repository.*

6 Reconstruction

We reconstruct the printout model, represented as a polygonal mesh, in two steps: (1) generate a hallowed surface mesh according to the composite skeleton graph and the width function defined on it; and (2) reconstruct a printable mesh with specified thickness via an offsetting operation.

6.1 Surface Mesh Reconstruction

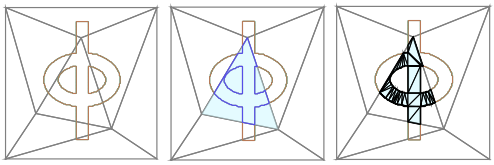


Figure 11: *Pattern baking: (a) The input contours, (b) The boolean intersection between a triangle and the input contours, and (c) Constrained Delaunay triangulation of the intersecting region.*

Based on the skeleton graph and the width function, we can recover a set of contours $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$ bounding the solid area that should be kept; See Figure 11a. For the i-th triangle T of the base triangle mesh, we perform the 2D boolean intersection between T and the composite contour $\mathcal{C}$; See Figure 11b. After that, we build the constrained Delaunay triangulation of the region $T \bigcap \mathcal{C}$ and get a collection of subtriangles $\mathcal{T}_i = \{T_1 \in T, T_2 \in T, \cdots, T_{n_i} \in T\}$; See Figure 11c. Performing the boolean operation all over the base mesh yields a triangle pool $\bigcup_i \mathcal{T}_i$ that actually gives the baked mesh $\mathcal{M}_b$ defined by the synthetic pattern elements. Note that $\mathcal{M}_b$ serves as both the input of stress analysis and the base mesh for generating a printable mesh with specified thickness.

6.2 Final Mesh Generation

Suppose that a thickness function $\tau(\cdot)$ has been specified on a surface $\mathcal{M}_b$. We compute the bounding surface $\mathcal{M}_b^\tau$ of the volume induced from the pair $(\mathcal{M}_b, \tau)$. For a point p in $\mathbb{R}^3$, there always exists a closest point $q \in \mathcal{M}_b$. We say p lies inside the volume $\mathcal{M}_b^\tau$

if and only if

$$\|p - q\| \leq \frac{\tau(q)}{2} \tag{6}$$

and

$$\frac{(p - q) \cdot \text{Normal}(q)}{\|p - q\|} = 1, \tag{7}$$

where $\text{Normal}(q)$ is the surface normal at q. Equation (7) serves only when q is located on the open boundary of $\mathcal{M}_b$. Or alternatively, we can use

$$F(p) \triangleq \|p - q\|^2 - \frac{\tau(q)}{2}(p - q) \cdot \text{Normal}(q) \leq 0 \tag{8}$$

to define the volume of $\mathcal{M}_b^\tau$. Sometimes we prefer inflating the point q to all directions, rather than only along $\text{Normal}(q)$, when q is located on the open boundary. And in this case, $F(p)$ reduces to

$$F(p) \triangleq \|p - q\|^2 - \frac{\tau^2(q)}{4} \leq 0. \tag{9}$$

We use the Poisson surface reconstruction [Kazhdan and Hoppe 2013] to extract the triangle mesh approximating the surface $F(p) = 0$, which is the boundary surface of the virtual model for fabrication.

7 Results

7.1 Basic Synthesis

We first present some synthesis results in 2D planar domain for validation. Figure 12 shows a variety of results generated by our method. As seen from the results, our algorithm generates smooth connections between patterns via either tangent connections or overlapping their similar features.

Since we formulate filigree synthesis as dense packing problem with appearance constraints, we compare our results with the state-of-art packing method [Hu et al. 2016] in Figure 13. As shown in Figure 13, conventional packing method cannot guarantee full connections between adjacent elements thus fails to satisfy the printable criteria. Moreover, the packing method tends to randomly place the element since it cannot exploit the partial shape similarity between elements to form smooth connections. Our method, on the other hand, is capable to ensure all the elements are connected in one piece while naturally join them without noticeable artifacts.

By allowing partial overlapping with good shape alignment, our method smoothly connects adjacent patterns of different shapes. Figure 14 demonstrates our synthesis results when multiple-class patterns are specified as input. Here, our method automatically detects the similar parts between different patterns and join them in a natural way. In Figure 1c, different filigree elements are used for synthesis on different parts of a vase. All these exemplar pattern elements are from actual filigree jewels. These results show that our method is capable of producing high-quality filigree decorations.

7.2 Control Field Editing

Our method can adapt the size and orientation of pattern elements according to an underlying control field on the base surface. Figure 15 shows different control fields on a 2D domain and the corresponding synthesized filigree pattern. Clearly, the pattern elements change their size and orientation according to the control field, while maintaining natural connections between adjacent elements.

Figure 16 shows a field-controlled synthesis result on a dress model. The control field here consists of an orientation field and a scaling field. We have developed a user interface to let the user specify the orientation field by sketching lines on the surface mesh, as shown in yellow curves in Figure 16a. We then generate a smooth orientation field by treating the sketch lines as constraints. Here the code from [Diamanti et al. 2014] is used to build the orientation field. The user can specify the scaling field by drawing closed regions on the base surface and assign scaling values to the designated regions. Then the scaling values are propagated to the entire domain via error diffusion.

Our method also supports boundary-aware synthesis in that we only synthesize a filigree pattern within a selected region. We observe that proper boundary handling is important in boundary-aware synthesis in order to produce satisfactory results.Hence, we add addi-

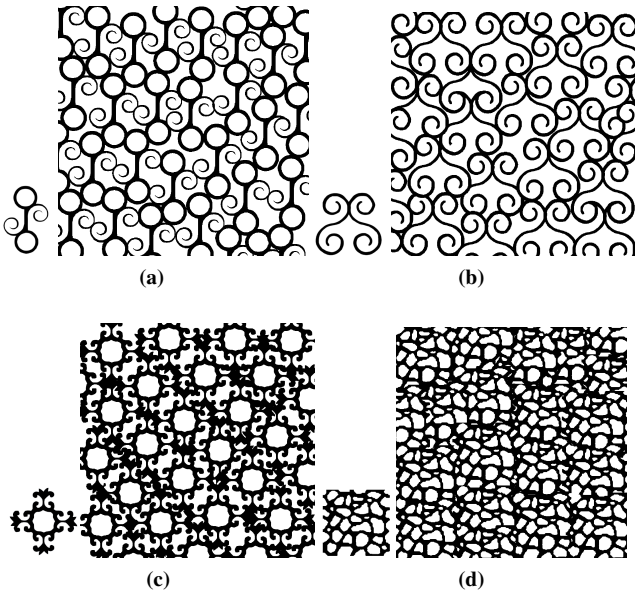


Figure 12: *Single-class synthesis results on 2D in which a variety of different elements are used as input.*

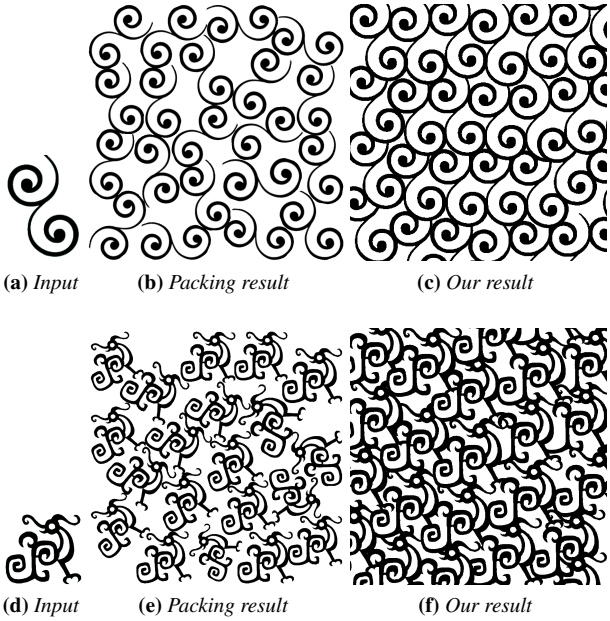
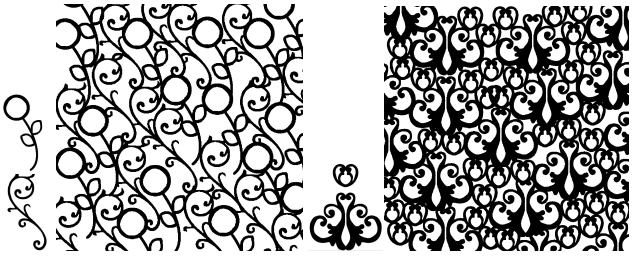
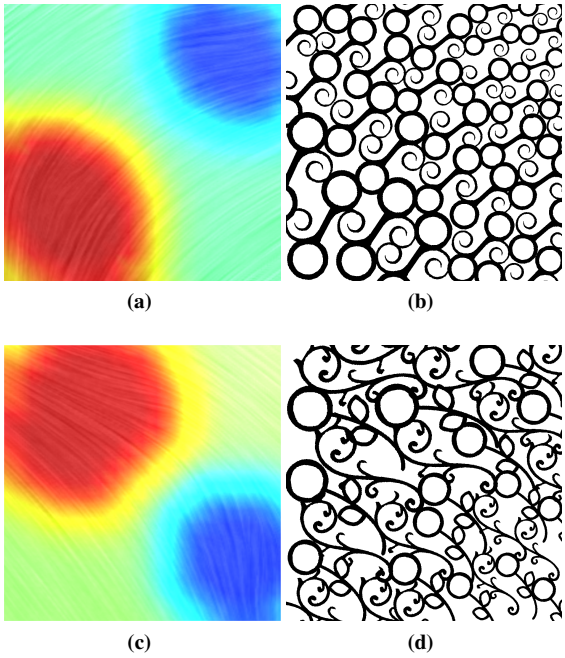


Figure 13: *Comparisons with the packing method in [Hu et al. 2016].*



(a) (b)

Figure 14: *Multi-class synthesis results on 2D planar domain.*



(a) (b)

(c) (d)

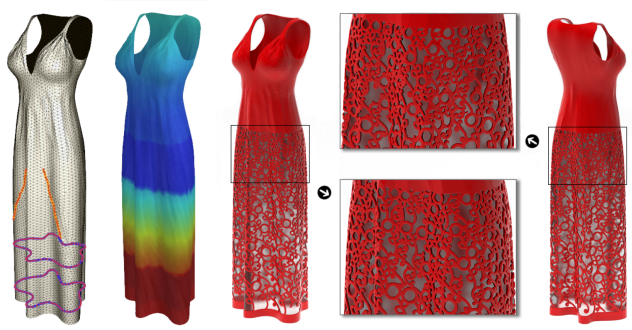
Figure 15: *Results of field controlled synthesis in 2D planar domain. For each row, the input control field is shown on the left. The control field is visualized using line integral convolution (LIC), where the color encodes the scaling of the elements (red - larger scale; blue - smaller scale) and the orientation is illustrated by streamlines.*

tional constraints to encourage pattern elements next to the domain boundary to have tangential contact with boundary curves. This can easily be achieved within our framework by applying proper translation and limited deformation during pattern synthesis.

7.3 Class Number Control

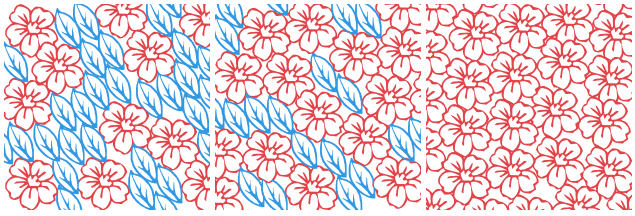
For multi-class synthesis, our method is capable of controlling the percentages of different classes of elements that appear in the final output. Figure 17 shows a series of results with an increasing number of flower patterns until it becomes a case of single-class synthesis. This is mainly achieved in the initialization step, in which the initial pattern distribution is generated. We adopt the class control strategy in [Wei 2010]. According to Equation (1) in [Wei 2010], the pattern with larger size will have lower priority to sample from. We control the percentage of each class by multiplying a scaling factor to the size of the pattern. If the scaling factor of an element is set to infinity, then that element will not appear in the output domain which makes single-class synthesis possible.

To achieve uniform distribution among different classes of elements, we penalize the overlap area if one element overlaps with an instance of the same element. Specifically, overlapping with the same class of elements will lead to higher overlapping ratio, thus will be rejected with higher probability. Figure 18 shows three examples of decorating the outer shell of a vase with different percentages of the flower and leaf patterns.



(a) (b) (c) *Synthesis result*

Figure 16: ***(a)*** *User interface. User can easily design the control field via sketching lines (yellow with red dots) on the surface mesh to define the orientation field and assigning scale values to hand-drawn (blue lines with red dots) regions to design the scaling field.* ***(b)*** *Control field. An example control field generated by user which is the input field for the synthesis result in* ***(c)****.* ***(c)*** *We support constrained synthesis within the user-specified regions. The synthesized elements are well aligned to the boundary and transformed according to the input control field. The input dress model is courtesy of Open3DModel.*



(a) (b) (c)

Figure 17: *2D Synthesis results with different percentages of each element class.*

(a) (b) (c)

Figure 18: *Synthesis results with different percentages of each element class in 3D.*

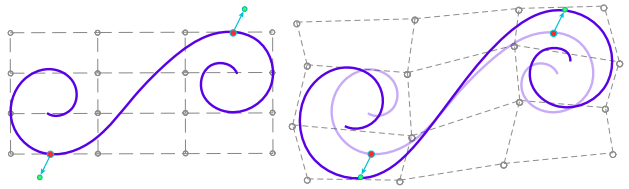
7.4 Fabrication

Figure 19 shows the printouts of our synthesis results. Note that the base elements remain easily identifiable on the surfaces while the

Figure 19: *Prototypes printed by Stratasys Fortus 400mc and EOS FORMIGA P 110.*

Parameters	Bunny	Lamp	Dragon Vase	Vase	Dress
σ_f	5.0%	4.5%	5.0%	5.0%	4.5%
σ_d	0.15	0.13	0.15	0.17	0.16

Table 1: *Parameters used in all tested models.*



(a) **(b)**

Figure 20: *(a) A 2D bicubic Bézier surface patch covers the skeleton, with its control points regularly distributed. The specified points (in red) must be moved to corresponding destinations (in green). (b) The soft constraints to avoid large distortions are also taken into account by moving the control points.*

element connections are smooth and robust even in highly curved regions, e.g. the bunny ears. The powder-based 3D printers present challenges on the model as the printout is very fragile when removed from the powder basin. However, our model is successfully printed with thin features, due to effective structural optimization.

8 Implementation and Performance

8.1 Input and Parameters

The input model of our method is a triangular mesh. All computations (e.g. modified Hausdorff distance) are done on a locally parameterized 2D surface patch. Each pattern has an up vector, which we maintain aligned with the underlying vector field to rotate elements along the surface. The patterns are mapped back to the surface mesh, so they naturally bend with the input model.

Our method is fully automatic but there are two main parameters that would affect the final results: (1) the threshold σ_f to filter element pairs based on matching quality. We typically use 5%. Larger values would lead to more randomization in the output; (2) the control of the distortion σ_d during deformations. A larger tolerance leads to larger deformations. We typically use 0.15. A complete list of the values employed for each model is shown in Table 1.

8.2 Deformation

For filigree synthesis problem, there is no strict requirement that the filigree element be kept in shape and size. Hence, we take the flexibility of slightly deforming filigree elements and develop non-rigid deformation method that is extensively used in our framework to enable smooth tangential contact and improve alignment of partial overlaps. Our method is fast because it needs only to solve a linear system of equations without iterative computation.

In the context of filigree synthesis, we need to deform a given pattern element P into another pattern shape P' with the following constraints: (1) Hard constraints. Some selected points p_k, $k = 1, 2, \ldots, m$, on P are mapped to some designated corresponding points p'_k on P'. (2) Soft constraints. The shape of P' is similar to that P as much as possible, not allowing a large distortion in any local or global subpart of the pattern element. Note that the deformation is applied to skeleton graphs, the internal representation of filigree patterns.

Our deformation method follows the spirit of 2D free-form deformation using a bicubic Bézier surface patch [Sederberg and Parry 1986] to deform the region containing a filigree element. Specifically, given a pattern element S to be deformed, we use a rectangular bounding region to cover P and define a bicubic Bézier surface patch $S^0(u, v)$ with its array of 4 by 4 control points p^0_{ij}, $i, j = 0, 1, 2, 3$, being regularly positioned within the bounding rectangular region, as shown in Figure 20. Since its control points p^0_{ij} are regularly distributed, $S^0(u, v)$ reduces to an affine mapping between the parameter space (u, v) and the output domain space (x, y) in which the filigree element S lies. Hence, for an arbitrary point $q^0_k = (x_k, y_k)$ of the output domain, it is easy to find its corresponding unique parameter values (u_k, v_k), with $q^0_k = S^0(u_k, v_k)$. With the fixed parameter values (u_k, v_k), the initial control points p^0_{ij} will be replaced by variable control points p_{ij} during the deformation process, so the point $q^0_k = (x_k, y_k)$ will be mapped to $q_k = S(u_k, v_k)$, where S is the new bicubic Bézier surface patch.

We now consider the following four types of constraints applied to the deformation.

Type 1. Suppose that we want to map some selected points q^0_k on the pattern P to their corresponding points q_k, $k = 1, 2, \ldots, m$. Then we have the constraints

$$a_k \equiv S(u_k, v_k) - q_k = 0, \quad k = 1, 2, \ldots, m. \tag{10}$$

Note that $S(u_k, v_k)$ is a linear combination of the unknown control points $\{p_{ij}\}$.

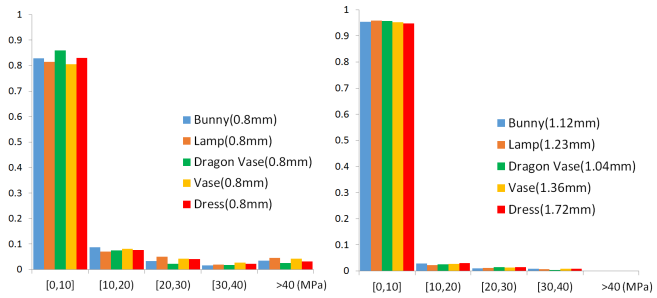
Type 2. We wish to keep the domain containing the pattern P to be deformed as little as possible. Therefore, we impose the "soft" constraint that any three consecutive control points that are equally spaced in the initial setting to still keep being equally spaced. Let $p_{i,j}$, $p_{i',j'}$ and $p_{i'',j''}$ denote three such control points, with p_{ij} lying between the other two. Then this constraint is expressed as

$$c_{ij} \equiv p_{i,j} - \frac{1}{2}(p_{i',j'} + p_{i'',j''}) = 0. \tag{11}$$

Type 3. Recall that the skeleton graph of a filigree pattern is represented by polygonal curves. To keep the shape and orientation of the skeleton graph under deformation, we first require that each straight-line segment of the skeletal element keep its original direction as much as possible. Let v^0_i, v^0_j be two consecutive vertices of the skeleton graph before deformation and let v_i, v_j be their corresponding points after deformation. Then we impose

$$d_{ij} \equiv (v_i - v_j) - \tau_{ij}(v^0_i - v^0_j) = 0 \tag{12}$$

where τ_{ij} is the scaling factor between the two parallel vectors. Note that the points v_i and v_j are not new variables but are expressed as functions of the variable control points p_{ij}. However, the scaling factors are new variables and they need to be regularized as follows to prevent excessive shape distortion of the skeleton graph.



(a) *Before structural optimization* **(b)** *After structural optimization*

Figure 21: *Stress distribution of all tested models before and after structural optimization.*

Type 4. We assume that the scaling factors $\{\tau_{ij}\}$, imposed on the segments of the skeletal element, smoothly change on the domain of interest. Suppose the segment $\overline{v_i^0 v_j^0}$, with a scaling factor τ_{ij}, has h neighboring segments, i.e. sharing an endpoint with $\overline{v_i^0 v_j^0}$, whose corresponding scaling factors are $\tau_{ij}^1, \tau_{ij}^2, \cdots, \tau_{ij}^h$, we have

$$g_{ij} \equiv \tau_{ij} - \frac{1}{h}(\tau_{ij}^1 + \tau_{ij}^2 + \cdots + \tau_{ij}^h) = 0. \quad (13)$$

Then we form a constrained linear least squares problem by including the hard constraints of type 1 with Lagrange multipliers, and the other soft constraints of types 2, 3 and 4 using penalty terms. Let S denote the set of control points p_{ij} and T denote the set of scaling factors τ_{ij}. Then the resulting objective function to minimize is

$$F(S,T) = \sum_k \theta_k a_k + \lambda_1 \sum_{ij} c_{ij}^2 + \lambda_2 \sum_{ij} d_{ij}^2 + \lambda_3 \sum_{ij} g_{ij}^2 \quad (14)$$

where θ_k are Lagrange multipliers for the constraints $a_k = 0$ of type 1, and $\lambda_1, \lambda_2, \lambda_3$ are weighting coefficients of the other penalty terms. In our implementation we take $\lambda_1 = 2, \lambda_2 = 60$ and $\lambda_3 = 0.5$. Because $F(S,T)$ is quadratic with regard to the variables S and T with linear constraints $a_k = 0$, it can be minimized efficiently by solving a linear system of equations with a sparse coefficient matrix.

8.3 Quantitative Analysis

8.3.1 Stress

We compare the stress distribution of all input models before and after structural optimization in Figure 21. The horizontal axis shows the intervals of Von Mises stress while the vertical axis is the percentage of nodes lying in each interval. The yield stress of the plastic material used in our models is 40MPa. As seen from Figure 21a, before structural optimization, there exist weak nodes (stress larger than 40MPa) in resulting models. However, no weak nodes are detected after optimization thanks to stronger pattern connection and increase of model thickness (as shown in Figure 21).

8.3.2 Matching Energy

Figure 22 shows the energy curve of appearance optimization with respect to iterations. The y axis is the pattern matching energy normalized by pattern number. Note that each iteration contains both placement optimization (Section 4.3) and boosting (Section 4.4). The matching energy generally decreases as the number of iterations increases, thus gradually improve the appearance. The jumps seen in the curves are due to new pattern connections added in structural optimization. The iterations would stop if the stopping criteria (Section 4.4) is satisfied. Normally, models covered by less and simpler patterns will converge faster. As seen from Figure 22, all tested models are stably converged to low PME values.

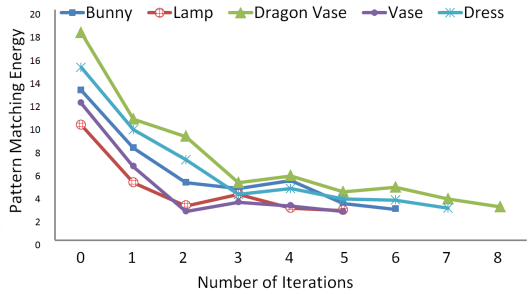


Figure 22: *Energy curve of appearance optimization of all input models.*

2D Result	#Ele	#Iter	t_{total}	2D Result	#Ele	#Iter	t_{total}
Fig. 12a	31	2	10.2	Fig. 14a	34	2	13.7
Fig. 12b	25	2	6.6	Fig. 14b	57	3	25.3
Fig. 12c	28	2	8.3	Fig. 15b	32	2	11.3
Fig. 12d	20	2	11.4	Fig. 15d	32	2	11.9
Fig. 13c	28	2	7.9	Fig. 17a	41	2	16.2
Fig. 13f	23	2	7.3	Fig. 17b	35	2	14.2
Fig. 17c	26	2	8.0				

Table 2: *Statistics of all the 2D synthesis results shown in the paper. The iteration number refers to the number of iterations for pattern synthesis, including both placement optimization and boosting. The timings are in seconds.*

3D Model	# V	#F	# E	$\#I_{syn}$	$\#I_{sct}$	t_{syn}	t_{sct}
Bunny	57154	114304	242	6	2	630	186
Lamp	4679	9098	75	5	2	138	36
Dragon Vase	29891	59786	521	8	3	1182	246
Vase	5419	10787	129	5	2	252	72
Dress	8172	16117	316	7	2	792	198

Table 3: *Statistics of some 3D synthesis results, showing the number of vertices, the number of faces, the number of elements, the number of iterations of pattern synthesis, the number of iterations of structural optimization, timings for pattern synthesis, and timings for structural optimization (in seconds).*

8.4 Timing

Table 2 summarizes the statistics of some synthesis results in 2D. When the number of patterns runs between 20 and 60, all the results are produced within 3 iterations. The computing time of 3D synthesis results are shown in Table 3. Unlike the 2D case, most of the computation time is spent on computing the local parameterizations of local surface patch. Thus, the computing time is highly related to the number of vertices and faces on the mesh surface. The results reported above were produced on a PC with Intel i7-4770 CPU and 16GB RAM.

9 Conclusion

We have proposed a new method for filigree synthesis based on a Pattern Matching Energy (PME) function, equipped with a stochastic optimization strategy, to guarantee that the output model is sufficiently connected, visually artifact-free and structurally strong for fabrication. Our method also allows for flexible user controls, such as the scale and orientation of pattern elements on the base surfaces.

Our method, in its current form, cannot handle those pattern elements that lack an obvious skeletal representation, e.g. the chessboard texture, or the case where a good local alignment between the input pattern elements cannot be found. In addition, the optimization phase is still time consuming due to the mesh reconstruction required by mechanical simulator. Future works include how

to support a greater variety of pattern types and investigation into the possibility of conducting structural optimization directly on the skeleton graph.

Acknowledgements

We would like to thank Yating Yue for helping prepare input filigree elements and anonymous reviewers for their insightful comments. We would like to thank LibIGL [Jacobson et al. 2016] for providing a nice toolbox of geometry processing. The work is supported by the Hong Kong GRF (HKU 717813E), NSFC (61272019, 61572021, 61332015, 61300168) and ERC ShapeForge (StG-2012-307877). The inset photos in introduction are cropped from https://pixabay.com/en/pocket-watch-jewel-chain-stone-560937/ and https://pixabay.com/en/medallion-necklace-metal-flower-1106635/ (courtesy of Yummymoon and CircoD respectively).

References

BATHE, K.-J. 2006. *Finite element procedures*. Klaus-Jurgen Bathe.

CHENG, D., AND WONG, K., 2009. 3doodler seashell pattern lace plastic dress. http://www.s-h-i-g-o.com/#!project/c147c.

DIAMANTI, O., VAXMAN, A., PANOZZO, D., AND SORKINE-HORNUNG, O. 2014. Designing n-PolyVector fields with complex polynomials. *Computer Graphics Forum 33*, 5, 1–11.

DUBUISSON, M. P., AND JAIN, A. K. 1994. A modified Hausdorff distance for object matching. In *Pattern Recognition*, vol. 1, 566–568.

DUMAS, J., LU, A., LEFEBVRE, S., WU, J., AND DICK, C. 2015. By-Example Synthesis of Structurally Sound Patterns. *ACM Transactions on Graphics (TOG) 34*, 4.

HARKER, J., 2011. Crania Anatomica Filigre: Me to You. https://www.kickstarter.com/projects/joshharker/crania-anatomica-filigre-me-to-you.

HU, W., CHEN, Z., PAN, H., YU, Y., GRINSPUN, E., AND WANG, W. 2016. Surface mosaic synthesis with irregular tiles. *IEEE Trans. Vis. Comput. Graph. 22*, 3, 1302–1313.

HURTUT, T., LANDES, P.-E., THOLLOT, J., GOUSSEAU, Y., DROUILLHET, R., AND COEURJOLLY, J.-F. 2009. Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering*, ACM, 51–60.

JACOBSON, A., PANOZZO, D., ET AL., 2016. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

KAZHDAN, M., AND HOPPE, H. 2013. Screened Poisson surface reconstruction. *ACM Transactions on Graphics (TOG) 32*, 3, 29.

LEFEBVRE, S., AND HOPPE, H. 2005. Parallel controllable texture synthesis. *ACM Transactions on Graphics (TOG) 24*, 3, 777–786.

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics (TOG) 25*, 3, 541–548.

LI, Y., BAO, F., ZHANG, E., KOBAYASHI, Y., AND WONKA, P. 2011. Geometry synthesis on surfaces using field-guided shape grammars. *Visualization and Computer Graphics, IEEE Transactions on 17*, 2, 231–243.

MA, C., WEI, L.-Y., AND TONG, X. 2011. Discrete element textures. *ACM Transactions on Graphics (TOG) 30*, 4, 62:1–62:10.

MARTÍNEZ, J., DUMAS, J., LEFEBVRE, S., AND WEI, L.-Y. 2015. Structure and appearance optimization for controllable shape design. *ACM Transactions on Graphics (TOG) 34*, 6, 229:1–229:11.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '00, 465–470.

RICCIOTI, R., 2013. Mucem museum concrete filigree. http://www.lemayonline.com/en/wow/mucem-photographed-by-edmund-sumner.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *ACM SIGGRAPH computer graphics*, vol. 20, 151–160.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: Improving structural strength of 3D printable objects. *ACM Transactions on Graphics (TOG) 31*, 4, 48:1–48:11.

TURK, G. 2001. Texture synthesis on surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, 347–354.

UMETANI, N., AND SCHMIDT, R. 2013. Cross-sectional structural analysis for 3D printing optimization. In *SIGGRAPH Asia 2013 Technical Briefs*, 5:1–5:4.

WEI, L.-Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, 355–360.

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. In *Eurographics '09 State of the Art Report*, 93–117.

WEI, L.-Y. 2010. Multi-class blue noise sampling. *ACM Transactions on Graphics (TOG) 29*, 4, 79.

ZEHNDER, J., COROS, S., AND THOMASZEWSKI, B. 2016. Designing structurally-sound ornamental curve networks. SIGGRAPH 2016, to appear.

ZHANG, J., ZHOU, K., VELHO, L., GUO, B., AND SHUM, H.-Y. 2003. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics (TOG) 22*, 3, 295–302.

ZHOU, K., HUANG, X., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2006. Mesh quilting for geometric texture synthesis. *ACM Transactions on Graphics (TOG) 25*, 3, 690–697.

ZHOU, Q., PANETTA, J., AND ZORIN, D. 2013. Worst-case structural analysis. *ACM Transactions on Graphics (TOG) 32*, 4, 137:1–137:12.

ZHOU, S., JIANG, C., AND LEFEBVRE, S. 2014. Topology-constrained synthesis of vector patterns. *ACM Transactions on Graphics (TOG) 33*, 6, 1–11.

ZIENKIEWICZ, O. C., AND TAYLOR, R. L. 2005. *The finite element method for solid and structural mechanics*. Butterworth-Heinemann.