Man-made by Computer: On-the-Fly Fine Texture 3D Printing

Xin Yan
yanxin91@mail.sdu.edu.cn
Shandong University
Qingdao, Shandong, China

Lin Lu*
llu@sdu.edu.cn
Shandong University
Qingdao, Shandong, China

Andrei Sharf
asharf@cs.bgu.ac.il
Ben-Gurion University
Beer-Sheva, Israel

Yu Xing
Shandong University
Qingdao, Shandong, China

Yulu Sun
Shandong University
Qingdao, Shandong, China

Figure 1: 3D printed squirrel textured by our G-code modifications with variations of cracks simulating random cracks in a natural material.

ABSTRACT

Applying textures to 3D models are means for creating realistic looking objects. This is especially important in the 3D manufacturing domain as manufactured models should ideally comprise a natural and realistic appearance. Nevertheless, natural material textures usually consist of dense patterns and fine details. Their embedding onto 3D models is typically cumbersome, requiring large processing time and resulting in large size meshes. This paper presents a novel approach for direct embedding of fine scale geometric textures onto 3D printed models by *on-the-fly modification of the 3D printer's head*. Our idea is to embed 3D textures by revising the 3D printer's G-code, i.e., incorporating texture details through modification of the printer's path. Direct manipulation of the printer's head movement allows for fine-scale texture mapping and editing on-the-fly in the 3D printing process. Thus, our method avoids the computationally expensive texture mapping, mesh processing and manufacturing preprocessing. This allows embedding detailed geometric textures of unlimited density which can model manual manufacturing artifacts and natural material properties. Results demonstrate that our direct G-code textured models are printed robustly and efficiently in both space and time compared to traditional methods.

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**.

KEYWORDS

G-codes, geometric texture embedding, meshless

ACM Reference Format:
Xin Yan, Lin Lu, Andrei Sharf, Yu Xing, and Yulu Sun. 2021. Man-made by Computer: On-the-Fly Fine Texture 3D Printing. In *Symposium on Computational Fabrication (SCF '21), October 28–29, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485114.3485119

1 INTRODUCTION

3D printing techniques are rapidly expanding the possibilities of how physical objects are fabricated, enabling fabrication of physical objects and structures not previously possible [Gürsoy 2018]. Research into the technology of FDM printing (e.g., header's movement speed, extrusion amount and height, etc.) has allowed to create novel effects such as hair [Laput et al. 2015], micro-pillar structures [Ou et al. 2016] and fluffy textures [Takahashi and Miyashita 2017] without the need for any hardware innovations. The control and accuracy provided by these technologies ease the transition from digital to physical.

Despite these advantages, digital fabrication has yet addressed the promotion of realism and natural artifacts in 3D printed objects. Attention is mostly directed to how closely fabricated objects resemble their digital counterparts, rather than considering variations and artifacts due to human craftsmanship, manual manufacturing

*Corresponding author.

SCF '21, October 28–29, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9090-3/21/10…$15.00
https://doi.org/10.1145/3485114.3485119

and material natural phenomena. As stated in [Gourdoukis 2015]: "while digital fabrication tools offer the chance to reconnect with materiality and craftsmanship they are taking away the properties of unpredictability and uniqueness that are inherent in processes that are harnessing materiality".

In this work, we present a 3D printing technique that can incorporate fine geometric textures onto printed models in a direct manner through modifications of the printer's head movement. This allows embedding detailed geometric textures of unlimited density which can model manual manufacturing artifacts and natural material properties. Our technique avoids the need for computational expensive texture mapping and 3D fabrication preprocessing. Instead, the 3D geometric texture is embedded onto the model on-the-fly, through re-programming of the 3D printer's G-code commands.

Textures have been extensively explored in computer graphics as means of generating realistic-looking models [Zhou et al. 2006]. In our context, textures were showed successfully in capturing complex natural phenomena [Bellini et al. 2016]. Texture synthesis for curved surface from exemplar has drawn a lot of research works [Wei et al. 2009]. Nevertheless, their application to 3D objects requires high-resolution meshes and large computation time. Textures have also been used for 3D printing of meshes with different geometric effects [Chen et al. 2016; Dumas et al. 2015; Liu et al. 2020; Yang et al. 2019; Zehnder et al. 2016]. In these works, 3D geometric textures are first mapped onto the mesh surface which is then sent for slicing and path planning (3D printing preprocessing). Typically, this is a time-consuming step governed by the mesh size, amount of fine detail and geometric complexity of the model.

Our method operates at G-code level and directly embeds geometric textures on the object's surface. The pipeline consists of a G-code parser that computes and modifies contour layers that form the surface. This consists of computing consistent local mappings and vertex offsets between contours and texture. 3D printing is realized through modification of the 3D printer head motion path to account for the new contour offsets.

Our method allows direct control on the texturing process at printer level. Specifically, the user can add variations to the model by on-the-fly additional modification of the printer's head motion path (e.g., random, noise pattern, textures). Thus, we can simulate man-made manufacturing artifacts. We also allow users to control and locally modify 3D texture generation. By providing users with such tools and interfaces we put the user in the manufacturing loop allowing humans yet again to "reconnect with materiality and craftsmanship". Our method makes the following contributions:

- A novel texture embedding approach by modification of the extrusion path.
- It performs direct on-the-fly manipulation of texture geometries at printer level allowing novel texture patterns that can simulate handmade manufacturing artifacts.
- Texture is applied on G-code contours without requiring additional mesh processing. It generalizes to any shape as it is not sensitive to geometrical complexity.
- Since it does not operate on the mesh, it does not introduce any storage footprint and is printer-friendly.

2 RELATED WORK

Textures have been previously considered in the context of digital fabrication as means of enriching 3D printed models with geometric patterns and details. In the following, we review related works in this area as well as the manipulations and modification of 3D printer motion.

Fabrication aware geometric texture synthesis. Texture synthesis approaches have been adapted to 3D manufacturing by converting image textures into printable 3D structures. In [Dumas et al. 2015] 2D image exemplars are converted into 3D patterns by considering solid and empty pixels. They define a by-example texture synthesis approach, connecting the solid parts into a structurally sound 3D texture. Similarly, shapes consisting of structural patterns were optimized in [Martínez et al. 2015] to satisfy specific loading scenarios while preserving similarity to an exemplar pattern.

Filigrees were synthesized in [Chen et al. 2016] to generate 3D printable textures and delicate patterns. Similarly, spline curves were used as decorative surface patterns in [Zehnder et al. 2016]. The method performs structural analysis and lets a curve network deform and connect under user assistance, thus strengthening in corresponding regions. [Suzuki et al. 2017] proposes a tool that allows the user to interactively sketch and manipulate textures that are automatically converted into 3D printable patterns onto arbitrary surfaces. InfoTexture [Zhang et al. 2017] has been introduced as a computer-aided fabrication approach that combines 3D structures with the surface texture. The method decomposes a digital model into basic texture pieces tiled onto a 2D plane and assembled into 3D forms after fabrication. [Liu et al. 2020] models fabricable Escher dual-shape tilings from user-defined shapes. Similar to us, [Fu et al. 2018] decorates 3D models in a direct manner without the need for a local/global parameterization. Using diffusion curves on arbitrary meshes denoted Poisson Vector Graphics, their method allows 3D decorations of detailed geometries while minimizing the overall computational expenses.

OpenFab [Vidimče et al. 2013] is introduced for solving the high resolution and large data in multi-material 3D printing. Their pipeline also supports easily defining texture on the input model. Due to the printer resolution and physical properties of the material, fine details could not be preserved after fabrication. [Pintus et al. 2010] proposed a geometry enhancement technique that simulates the printing process and edits the input mesh to improve the readability of the printed object.

Outside the academic field, slicer software in the industry also tries to facilitate texture in fabrication. Cura software [Cur 2020] allows for adding white noise texture during the slicing procedure. Velocity Painting [Wheadon 2021] allows users to add image texture on a specified region.

In a different direction, 3D geometric texture manufacturing was utilized to explore tactile surface properties, haptics, and perception. HapticPrint was introduced in [Torres et al. 2015] to enable physical haptic design in 3D printed objects. Their method allows for defining and editing exterior geometric textures and interior stiffness constraints, yielding tactile (touch) and kinesthetic (pressure) characteristics. [Dassen and Bruns Alonso 2017] uses material sketches to design a set of haptic interaction prototypes to explore push button and knob haptic sensations. [Yasu 2017] proposes a method

for designing magnetic fields and tactile textures using digital plotting. Specifically, they write magnetic patterns on magnetic rubber sheets, which display unique haptic stimuli when rubbed. [Tymms et al. 2018] explores perceptual surface roughness by building a quantitative model that maps between texture geometry and tactile roughness. [Ion et al. 2018] presents metamaterial textures inspired by origami and surface wrinkling. It consists of 3D printed grid cells that fold when compressed by an external force. The tactile and appearance of texture are often linked. Changing the visual appearance would, in turn, affect the tactile feel. [Tymms et al. 2020] proposes a neural network-based optimization method to control the tactile property while maintaining its visual appearance.

These methods explore various geometric patterns and haptic properties of 3D printed surfaces. Nevertheless, their 3D textures are simple in terms of geometric detail and resolution. Our G-code level direct 3D texture printing method allows applying complex and dense geometric textures that directly represent materials properties and natural phenomena onto 3D printed models.

3D printer manipulation. Investigations into 3D printing parameters (e.g., header's movement speed, extrusion amount, and extrusion height) have extended the capability of 3D printers, allowing users to create novel 3D effects in addition to solid 3D objects without the need for any hardware innovations.

Dual-color mixing was introduced in [Reiner et al. 2014] as a dithering technique for generating continuous tones using two heads FDM 3D printers. Authors modify the head movement to introduce small geometric offsets between layers, producing interleaved color patterns that convey an impression of smooth color blending. [Song et al. 2019] makes colored FDM printing possible by dividing layer further into a set of strata, each stratum has a constant color. By locally changing the thickness of the stratum, they could change the perceived color at the given location.

Inspired by the glue gun stringing behavior, a 3D hair printing technique was introduced in [Laput et al. 2015]. The method allows fabricating soft strands and fibers by manipulating the FDM printer head extrusion amount, feed rate, and running distance. Similarly, Cilllia was introduced in [Ou et al. 2016] for hair-like structures fabrication in stereolithography (SLA) 3D printing. The method generates printable layers represented by bitmap texture that contain dense and complex hair structures. Similar to us, they map hair arrays onto the object's surface by directly modifying the 3D printer's slicing protocol. Nevertheless, their method focuses on textures representing specific cone-like hair geometries, while our method aims at general geometric textures that represent natural material phenomena.

The control of extruded material in fused deposition modeling (FDM) 3D printers was previously explored. In [Takahashi and Miyashita 2016], authors increase and decrease the amount of material extruded during printing, yielding aesthetic pattern sheets and tactile objects. Similarly, the relation between the height position of the extruder and the amount of extruded material is explored in [Takahashi and Miyashita 2017] to enhance the capability of new inherent expressions for FDM 3D printers. [Kuipers et al. 2018] introduces a novel halftoning technique for fabricating 3D grayscale objects using the FDM 3D printing method. Their method modulates the visible width of printed lines of two alternating colors to produce continuous-tone gradients. Finally, 3D printed fabric is introduced in [Takahashi and Kim 2019] for fabricating soft and flexible textiles using an FDM 3D printer. Authors control the movement of the printer header to alternately weave stringing fibers across a row of pillars and generate a thin sheet of fabric.

Our method takes a step in this direction and allows embedding geometric textures of high resolution and complexity directly onto the surface through modulation of the printing toolpath.

3 OVERVIEW

Given a 3D shape, our aim is to embed a given 3D texture onto the shape's surface by directly modifying the printer's head motion path. We assume a G-code file corresponding to the original 3D shape is given. Since we focus on extrusion-based printing techniques, the input G-code file consists of a decomposition of the 3D shape into a set of contour layers which govern the printer's head movement. We utilize the G-code contours representation and compute a mapping between the 3D texture and contours, layer-by-layer, while accounting for global structural coherence.

In the first step (see Figure 2 mid-left), we compute a Quad tessellation of the input surface. Quad tiles allow a simple mapping between texture exemplars and the surface. In order to handle complex topologies, we compute a Reeb graph on top of the contour layers in slicing direction. Thus we account for topological complexity and monitor topology singularities in the surface. We use Reeb graph and contours to evenly distribute Quad vertices on the surface and connect them in a topologically persistent manner. We allow users to control Quad tessellation resolution and orientation.

In the next step, we compute for each contour, a mapping between contour vertices and corresponding texture exemplar (defined by quad to texture mapping). This yields offsets of contour vertices according to their mapping into the 3D texture (see Figure 2 mid-right). We then apply a path planning to fill the interior of each contour and generate the new G-code commands with geometric texture embedded.

4 G-CODE QUAD TESSELLATION

In this step, we utilize the G-code file and contour layers decomposition to efficiently compute texture mapping information for any 3D shape. Essentially, we compute a Quad tessellation of the surface by processing contour layers extracted from the G-code file. Quads are then used as texture proxies which are directly mapped to texture tiles. We also compute a Reeb graph from the contour layers to detect topological singularities. We then sample the contours uniformly and connect sample points to form a valid Quad tessellation of the surface.

Reeb graph. Reeb graphs are compact shape descriptors that convey topological information related to level sets of a function on a shape [Reeb 1946]. They were successfully applied for topology-based shape decomposition, segmentation, and parameterization [Patanè et al. 2004; Zhang et al. 2005] We follow this path and compute a Reeb graph by considering the G-code slicing of the shape into contours as a simple height function in z up-direction.

We first extract contours for each slice layer from the G-code file. Then, we trace contours along the slicing direction and examine topological changes. Specifically, between two adjacent layers,

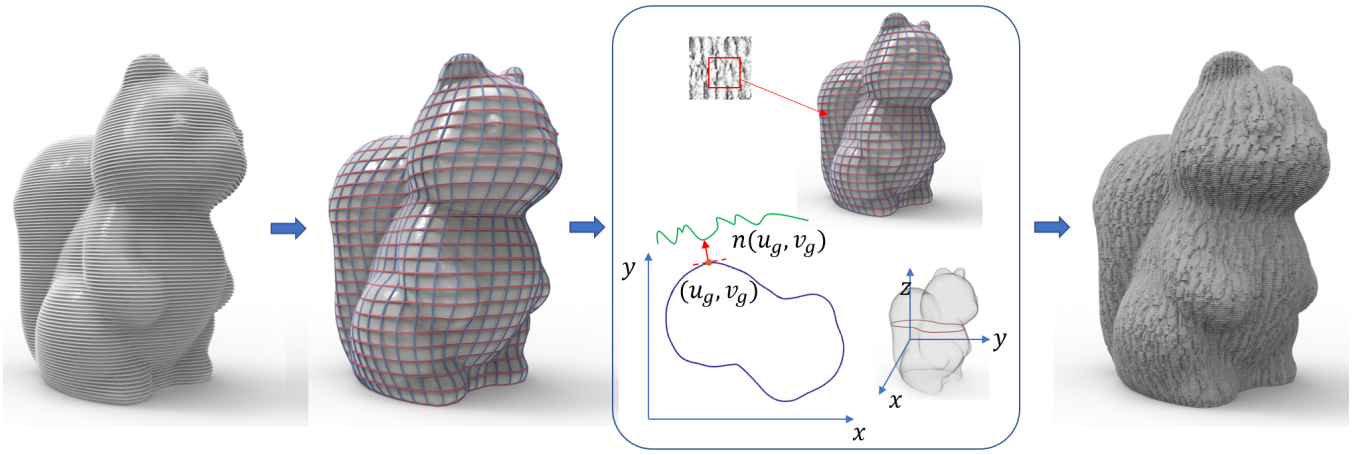

Figure 2: 3D texture printing pipeline. Given an input 3D shape (leftmost), we tessellate its surface into quads and map 3D texture tiles onto quads (mid-left). We compute vertex offsets between contours and texture and modify the printer's G-code and head movement (mid-right), resulting in a 3D printable textured shape (rightmost).

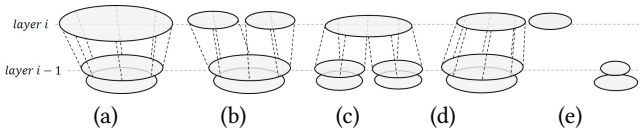

Figure 3: Contour relationships between adjacent layers: (left-to-right) maintain, split, merge, appear, disappear.

topological changes of contours include *split, merge, appear or disappear* (see Figure 3). To determine contour relationships between adjacent layers, we project the contour onto successor layers along the slicing direction. If it intersects one or more contours in the predecessor layer, then it follows cases a-c in Figure 3. If it has no intersection in the predecessor layer, then it is a new contour, case d in Figure 3. If it has no intersection with any contour in the successor layer, then it disappears, case e in Figure 3. We trace contours and their cases (i.e., critical points) and build a corresponding Reeb graph (see Figure 4).

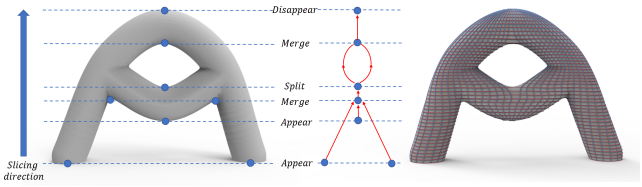

Figure 4: Tracing G-code contours along slicing direction (left), corresponding Reeb graph (middle) and Quad tessellation (right).

Surface sampling. Given the slicing up direction, we choose the bottom layer in the G-code as the base layer. Suppose the base layer has m independent contours, we evenly distribute k sample points on all m contours, where k is a user-defined parameter. We propagate the sampling points from the base layer, bottom to top, considering spatial coherence. During propagation, we project each sample point on a contour in layer $i-1$ to the closest point in layer i (successor layer). This projection holds for any of the contour cases: maintain, split, merge (Figure 3 a-c). In the case of a contour that appears (Figure 3 d), we treat it similarly to a new base layer and sample it evenly with the same sampling rate as the initial layer.

We follow two optimization steps during the point propagation between contours to ensure that the sampling points on the successor contour are distributed as smoothly as possible. Our optimization consists of two relaxation steps which we perform iteratively. First is uniformity: sample points should have an even distribution where the point with minimal projection distance is the anchor point. Second is minimal distance: sample points move such that the total distance between sample points and their projected counterparts in the predecessor contour is minimal (see Figure 5).

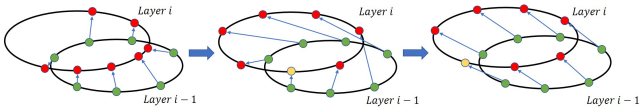

Figure 5: Propagation optimization of points from contour in layer $i-1$ to contour in layer i. Initially points are projected to a closest position (left), then evenly distributed from anchor point marked in yellow (middle), then moved together to minimize the projection error (right).

Quad tiling. Our sampling scheme yields a Quad tiling of the surface which is obtained by connecting neighboring sampling points along the contour direction (denoted by u) and in slicing (cross-layer) direction (denoted by v). Thus, a Quad is obtained by connecting together adjacent sample points along a contour (u) as well as points and their projected counterparts on adjacent layers (v).

Since projecting points connect across layers, their Quads may become distorted due to topological changes. Although we aim to

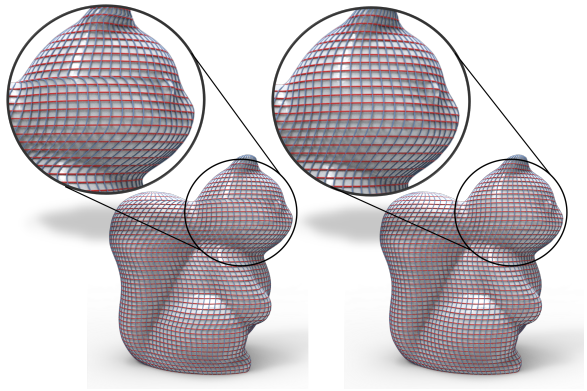

Figure 6: Initial Quad tiling (left) and smoothed (right).

minimize projection error between adjacent layers, significant topological changes along the slicing direction may yield non-smooth transitions (see Figure 6-left). Such distortions in the Quad tiling introduce visible distortions in the texture mapping. Hence, we apply a restricted Laplacian smoothing on the sample points and then project the points back onto their contours (see Figure 6-right).

(a) (b) (c)

Figure 7: A coarse Quad tiling of a vase (a) is refined by denser sampling (b) and deformed using a simple line curve guide (c) (in dotted square).

User control. Our method allows some design freedom in the texture mapping process. Specifically, the user may control the texture scale by defining the sampling density via k yielding different Quad tiling resolutions (see Figure 7 (a-b)). Thus, as Quad tiles are finer, texture details would be finer too (bounded by the initial texture resolution). Additionally, the users may guide the Quad tiling global orientation. This is achieved by drawing a cubic Bézier curve line that replaces the original slicing direction v. This affects the projection direction of points between adjacent layers and, in turn, the Quad tile orientation and texture position on the surface (Figure 7 (c)).

5 G-CODE TEXTURE MAPPING

Essentially, our Quad tessellation binds the surface with a coordinate space along the contour direction u and the slicing direction v. To map the texture to quads, a straightforward idea would be to assign the given texture to each Quad. Nevertheless, this would result in repetitive artifacts and boundary discontinuities (see Figure 8). To avoid boundary artifacts, we follow the texture synthesis method in [Efros and Freeman 2001].

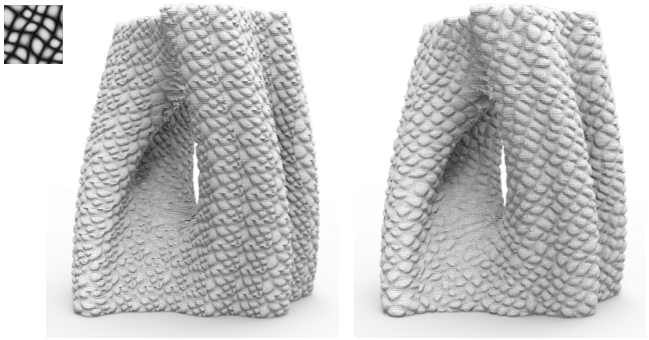

Figure 8: Direct texture mapping onto Quads yields artifacts at boundaries (left). Using texture synthesis optimization generates a smooth texture (right).

Since the surface is represented by a series of Quad tiles g, we denote the i^{th} Quad cell by g_i, and compute the optimal texture exemplar b_i following the image quilting method suggested in [Efros and Freeman 2001]. A texture binding is then obtained for each pair of (g_i, b_i).

Next, for a position (u_g, v_g) on a contour (u_k, v_k) and its corresponding tile g_i, its texture coordinates indicates a grayscale value in the geometric texture exemplar b_i. This grayscale value encodes a displacement value along the normal direction $n(u_g, v_g)$ in the contour layer (see Figure 2 (mid-right)). This allows offsetting the contour points and thus embed the texture onto the surface. Note that we super-sample each contour to match the texture resolution. The sampling rate is also bounded by fabrication resolution.

Procedural textures. Besides exemplar texture mapping, our framework also allows directly map procedural functions onto the G-code contours. Given any procedural texture function $f(u, v)$ we map it onto Quads successively. Similarly, after super-sampling of each contour, we locate its Quad cell and obtain its displacement value from its texture mapping to procedural texture for each sample on the contour. We then apply the displacement along the normal direction. Figure 11 demonstrates textures from Perlin noise functions with different parameters.

G-code file generation. A G-code file consists of machine setting commands and motion commands. The motion command moves the extruder to target positions and extrudes the proper amounts of materials to form the shape. In the final step of our pipeline, we convert the modified outer contour layers representing the textured surface back into G-code motion commands following a path plan. The path planning process generates the infill patterns and support structures based on the polygon contours and converts them to a toolpath. The toolpath patterns include raster, zigzag, contour-parallel, spiral, Fermat spiral, and other space-filling curves [Ding et al. 2014; Zhao et al. 2016].

For FDM printing with thermoplastics, we perform a hybrid fill: contour-parallel pattern to guarantee geometric accuracy of outer boundaries and a zigzag pattern of the interior filling for

each contour. Next, we calculate the printer's extrusion amount for each line in the G-code commands. We output each path in each layer in z increasing order. Our G-code format follows the CuraEngine [Cur 2020] and is compatible with most mainstream FDM 3D printers. Our method could also be easily integrated into slicers like Cura, which would make the geometric texture generation more convenient. Ceramics are of excellent mechanical properties and unique appearance. Due to the low-cost hardware, extrusion-based clay printing received increasing attention from industry and academia [Hergel et al. 2019]. For extrusion-based clay printing, the input models are mostly self-supporting shells, which conform to the pseudoplastic liquid characteristics. Thus, long as the displacement magnitude is restricted in the self-supporting area, the modulated single contour directly exported to G-code commands is good enough for fabrication (see Figure 15).

6 RESULTS

Our G-code texturing works on extrusion-based 3D printers. In the following, we generate fine geometric textures on various models, evaluate our performance, and compare them with existing standard texture mapping and 3D manufacturing methods. Furthermore, we validate our method by fabricating various models following our G-code files and using plastic and clay printing materials. Our method is implemented in C++ and run on an Intel Core^{TM} i7-6700K CPU @4.0GHz and 16GB RAM.

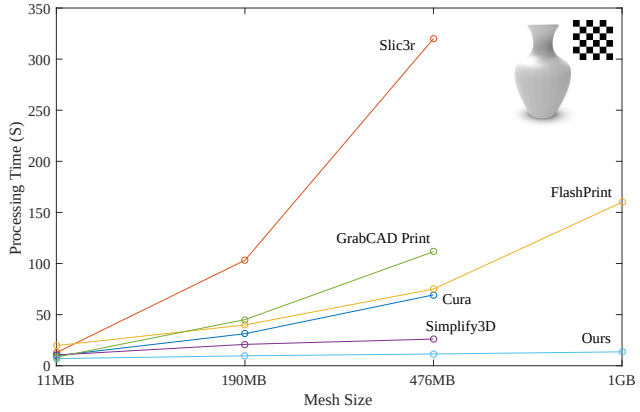

Figure 9: Performance comparison (processing time vs. mesh size) between 5 state-of-the-art 3D manufacturing tools and ours. Since our method does not have mesh representation during texture embedding, the mesh size here is an approximation by using the texture synthesised in our pipeline and perfrom texture baking in Maya® to get the corresponding mesh.

Performance. Our method embeds fine detail geometric textures directly into the G-code that is then 3D printed. Thus, our method does not add to mesh resolution and size while keeping low computational costs of texture embedding and G-code updating. In contrast, applying fine detail textures with traditional methods requires a high computational cost of texture synthesis and mapping. This is determined by the complexity of both input shape and texture. Additionally, the resulting textured mesh is very large due to its fine-detailed geometry. In 3D manufacturing, large meshes are a practical issue for data transmission and preprocessing. Specifically, their slicing and path planning processes would require extensive processing time and even may turn infeasible (due to memory or CPU limitations).

In Figure 9 we show a comparison between our method and commercial, state-of-the-art 3D printing tools that perform slicing and path planning on given meshes in terms of processing time and mesh size. The height of the tested model is 120mm. We set the layer height as 0.15mm with an infill density of 10% for all models. To examine the results considering possible detail decimation in slicing, we use checkerboard texture as the exemplar. Generally, all tools demonstrate a significant increase in processing times with the increase in mesh size. More critically, all but one tool (FlashPrint) could not handle meshes larger than 900MB. Here the mesh size only counts vertices and facets without additional information like normal or texture coordinates. Based on our experiments, the features are well preserved after slicing when the model size is not large, say less than 500MB. Most slicers do not work when the model size is large to 1GB, except for FlashPrint. However, FlashPrint simplifies the model during slicing, as shown in Figure 10, where we can see that geometric details are damaged after simplification. In contrast, our method avoids reconstructing a large texture mesh, and in fact, is insensitive to the textured shape complexity. Note the red line is almost flat in Figure 9.

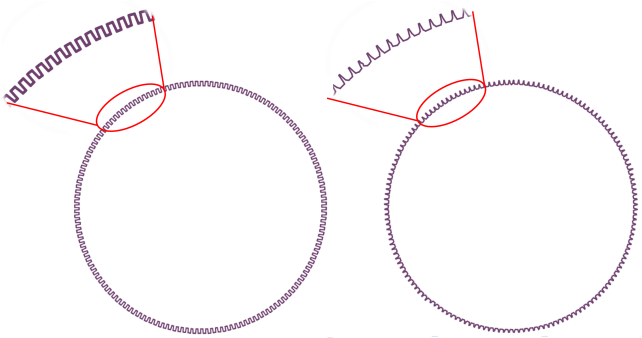

Figure 10: Cross-sections at the same height of a 1GB model with the same scale geometric textures. Left: our result is consistent with the original checkerboard texture; Right: the contour sliced by FlashPrint is of detail decimation.

User study evaluation. We have evaluated our on-the-fly texture embedding method against Maya® texturing tool. The purpose was to compare our method's direct texturing capabilities against another state-of-the-art mesh texturing tool (focusing just on texture mapping) in terms of processing time and data size.

We performed a short user study to measure processing times where we asked 4 different users (2 experts and 2 novices) to use both tools and apply 5 different textures on 4 different models. For our method, we record the time consumption from input a G-code model to output a G-code file with texture embedded. The processing time of embedding geometric textures on a model using Maya® includes importing the mesh, surface parameterization, texture displacement mapping, textured mesh generation, and exporting. We

provide a predefined Arnold® script that works in Maya® to facilitate the texture mapping and geometry baking operations. The users only need to adjust some parameters and output the resulting mesh.

The average processing time of all users is summarized in Table 1 (rows denote the 5 textures, columns denote the 4 different models). Our method has a clear advantage over using Maya®, which requires tedious work. Similarly, we compared textured mesh sizes when using Maya® texturing and our tool (Table 2). The G-code file size is significantly smaller since we do not need to reconstruct the fine-textured meshes.

Texturings. In this work, we have experimented with various textures that provide fine detail and complex geometries. When using procedural textures originating from functions and mathematical models, parameters controlling the texture generations can be changed to get very different results. Figure 1 shows variations of a crack geometric texture applied on a squirrel mode. This is achieved by directly editing the tiling orientation and resolution. In Figure 11 we demonstrate applying Perlin noise onto 3D models using the G-code contours. By changing the parameters in Perlin noise function, we obtain a family of textured shapes which resemble man-made manufacturing artifacts.

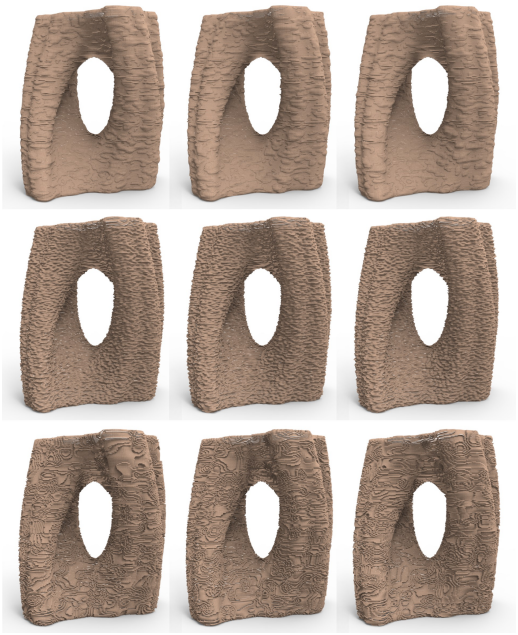

Figure 11: By changing the parameters in Perlin function we can get a family of textures.

To simulate material variations, we allow users to control the resolution and orientation of Quad tiles which affect the scale and shape of the final textured model. In Figure 12 top-row, we use the same input texture exemplar on two different Quad tiling resolutions of a stump model, yielding different texture details. In Figure 12 bottom-row, we change the Bézier curve guiding the Quad tiling orientation, yielding a different visual effect of the same base texture.

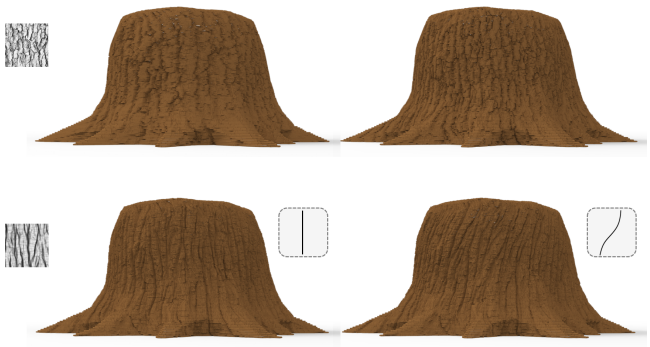

Figure 12: Given a texture exemplar, our system could generate different textures based on user guidance. Top-row shows different Quad-tiling resolution, bottom-row shows different v line affecting Quad-tiling orientation.

Texture Accumulation. A feature of our G-code direct texture embedding is that it is accumulative. I.e., we can re-use the output of our method as input to another texture embedding iteration. Thus, we can accumulate different texture patterns together using several iterations of our G-code texture embedding. This way, we can aggregate different textures together and generate complex texturing effects still on-the-fly in the 3D printing process. For example, users can get series of results mimicking some phenomena (see Figure 13).

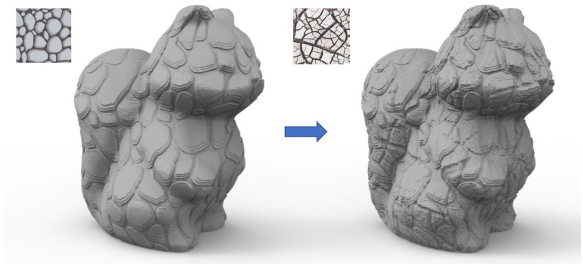

Figure 13: Example of texture accumulation. An initial stone texturing iteration (left) followed by adding a crack texturing iteration (right).

Fabrication evaluation. We fabricate some models on Hori-E5 desktop FDM printer (see Figure 14). We use Cura 15.4 to slice the digital model and get the G-code file. All models in our experiment are 75mm in height, and the layer height is 0.15mm. In Figure 14, left column is Marble-PLA and right column is Wood-PLA. These materials share similar physical properties as PLA material but contain marble and wood visual similarity. Each row consists of a model with two different textures. First two rows are models printed in "open surface mode" (printing the outer surface while keeping the top surface open). The last two rows are models in "normal mode" (printing with infill structures). Figure 17 shows the results for randomly disturbing the G-code commands by 20% and 30% of the displacement magnitude.

Table 1: Running time comparisons between Maya® texturing and our method (slicing time not included).

Model		Circle Vase			Triangle Vase			Squirrel			Bunny		
		Maya	Ours	Time Saving	Maya	Ours	Time Saving	Maya	Ours	Time Saving	Maya	Ours	Time Saving
Texture		360s	7.25s	97.9%	320s	6.12s	98.1%	400s	10.61s	97.3%	372s	8.3s	97.8%
		370s	7.17s	98.1%	360s	6.17s	98.3%	398s	10.42s	97.4%	380s	8.06s	97.9%
		485s	7.16s	98.5%	480s	6.15s	98.7%	520s	10.41s	98%	490s	7.98s	98.3%
		620s	7.19s	98.8%	600s	6.27s	99%	640s	10.48s	98.4%	620s	8.22s	98.7%
		380s	7.25s	98.1%	365s	6.21s	98.3%	390s	10.67s	97.3%	376s	8.23s	97.8%

Table 2: Data size comparisons between Maya® texturing and our method.

Model		Circle vase			Triangle vase			Squirrel			Bunny		
		Maya (MB)	Ours (MB)	Space saving	Maya (MB)	Ours (MB)	Space saving	Maya (MB)	Ours (MB)	Space saving	Maya (MB)	Ours (MB)	Space saving
Texture		611	8.30	98.6%	517	7.36	98.6%	995	8.65	99.1%	621	3.93	99.4%
		605	8.23	98.6%	522	7.42	98.6%	979	8.49	99.1%	633	3.79	99.4%
		604	8.19	98.6%	520	7.4	98.6%	982	8.75	99.1%	631	3.78	99.4%
		617	8.25	98.6%	530	7.54	98.6%	985	8.23	99.2%	615	3.68	99.4%
		612	8.31	98.6%	524	7.51	98.6%	962	7.92	99.2%	618	3.87	99.4%

We also verify our results on an extrusion-based ceramic printer, CERAMBOT Plus. The models are 200mm in height, and the layer thickness is 1mm. Because the layer thickness in ceramic printing is relatively large, if the z value in one layer is constant, there would be a seam where the z value changed. Changing the z value evenly in each layer and form a spiral printing path could solve this problem. Our method works well in this case. Figure 15 shows the spiral printing path of the vase model with Perlin noise, tree bark, and crack textures.

Except for extrusion-based printers, our method could be easily adapted to other printers. In Figure 16 we apply Perlin noise on a triply periodic minimal surface (TPMS) with intricate geometry. We use an AccuFab-D1 DLP (digital light processing) 3D printer for fabrication. The model is 80mm in height, and the printing resolution is 0.075mm in the x-y direction and 0.050mm in the z-direction.

DLP 3D printers use digital mask images to fabricate models and, therefore, a different G-code. We convert the contour layers into image masks using a scan line on the polygonal area filling algorithm. Then we set exposition times per image such that the whole model can be fabricated.

Limitation. Our method uses a direct geometric texture mapping on shape contours. I.e., we compute contour offsets from a mapped set of values. In cases where the shape has large concave regions or sharp edges, the resulting textured shape will consist of self-intersections when the texture encodes large offsets. These artifacts are common when adding geometric textures to a model. A possible solution is performing shape analysis to obtain an upper bound distance of the texture offset. Additionally, our texture mapping step (Quad tiling) does not handle shrinkage effects due to shape contours scaling down (see Figure 18 left). The texture offsetting operation only works in the x-y direction. As a result, in the top region of the half-sphere model, the amplitude of geometric texture gets smaller. Our method takes grayscale value to encode the offset distance to express the geometric texture without considering any perceptual aspects. Dealing with texture exemplar with dense feature like knitting (Figure 18 right), the resulted geometric texture would be indistinct.

7 DISCUSSION AND FUTURE WORK

We presented a novel approach for directly embedding fine-scale geometric textures onto 3D printed models by modifying the 3D printer's head. The manipulation of the printer's head movement allows fine-scale texture mapping and editing on-the-fly in the 3D printing process. The method avoids computationally expensive texture mapping, mesh processing, and manufacturing preprocessing. We show texturing of detailed geometries and modeling manual manufacturing artifacts and natural material properties using our method. Results demonstrate a large variety of robustly printed modified G-codes of textured models.

In the future, we plan to investigate more G-code manipulations for additional effects beyond the visual aesthetics of textures. For example, we plan to investigate functionalities that could be embedded onto different surfaces using G-code manipulation such as anti-slipping, velcro, etc. Additionally, we plan to extend our investigation of artifacts in manual manufacturing and establish a

Figure 14: FDM 3D printed textured models.

Figure 15: Ceramic 3D printed textured vases.

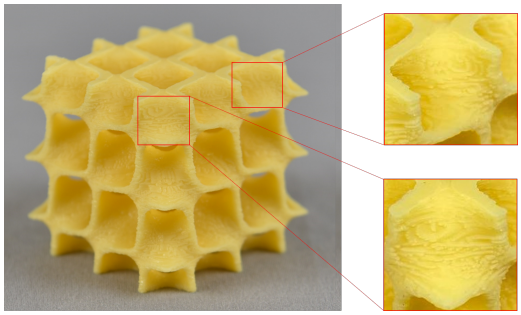

Figure 16: A 3D printed complex TPMS model textured with Perlin noise.

study to obtain a systematic model of such artifacts and relate them to the type of manual activity, material, tools, etc. This would help in further promoting realism in 3D manufacturing and advancing man-made by computer.

ACKNOWLEDGMENTS

We thank all the anonymous reviewers for their valuable comments and constructive suggestions. This work is supported by grants from NSFC (61972232). Special thanks to Prof. Daniel Cohen-Or for inspiring discussions on this project.

REFERENCES

2020. CuraEngine. https://github.com/Ultimaker/CuraEngine

Rachele Bellini, Yanir Kleiman, and Daniel Cohen-Or. 2016. Time-Varying Weathering in Texture Space. *ACM Trans. Graph.* 35, 4, Article 141 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925891

Weikai Chen, Xiaolong Zhang, Shiqing Xin, Yang Xia, Sylvain Lefebvre, and Wenping Wang. 2016. Synthesis of Filigrees for Digital Fabrication. *ACM Trans. Graph.* 35, 4, Article 98 (July 2016), 13 pages. https://doi.org/10.1145/2897824.2925911

Wendy Dassen and Miguel Bruns Alonso. 2017. Aesthetics of Haptics: An Experience Approach to Haptic Interaction Design. In *Proceedings of the 2017 ACM Conference Companion Publication on Designing Interactive Systems* (Edinburgh, United Kingdom) *(DIS '17 Companion)*. ACM, New York, NY, USA, 254–259. https://doi.org/10.1145/3064857.3079156

Donghong Ding, Zengxi Stephen Pan, Dominic Cuiuri, and Huijun Li. 2014. A tool-path generation strategy for wire and arc additive manufacturing. *The international journal of advanced manufacturing technology* 73, 1-4 (2014), 173–183.

Jérémie Dumas, An Lu, Sylvain Lefebvre, Jun Wu, and Christian Dick. 2015. By-example Synthesis of Structurally Sound Patterns. *ACM Trans. Graph.* 34, 4, Article 137 (July 2015), 12 pages. https://doi.org/10.1145/2766984

Alexei A. Efros and William T. Freeman. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*. ACM Press. https://doi.org/10.1145/383259.383296

Figure 17: The G-code commands are randomly disturbed by 20% (left) and 30% (right) of the displacement magnitude.

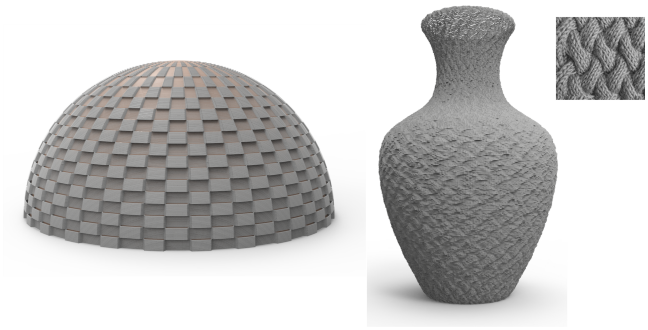

Figure 18: Left: Shrinkage limitation of our texture as contours get smaller towards the top of a half-sphere. Right: The resulted texture is indistinct for the knitting examplar.

Qian Fu, Fei Hou, Qian Sun, Shi-Qing Xin, Yong-Jin Liu, Wencheng Wang, Hong Qin, and Ying He. 2018. Decorating 3D models with Poisson vector graphics. *Computer-Aided Design* 102 (sep 2018), 1–11. https://doi.org/10.1016/j.cad.2018.04.019

Dimitris Gourdoukis. 2015. Digital craftsmanship: from the arts and crafts to digital fabrication. *ArchiDOCT* 2, 2 (2015), 43–57.

Benay Gürsoy. 2018. From Control to Uncertainty in 3D Printing with Clay. In *Proceedings of the 36th eCAADe Conference*, Vol. 2. Lodz, Poland, 21–30.

Jean Hergel, Kevin Hinz, Sylvain Lefebvre, and Bernhard Thomaszewski. 2019. Extrusion-Based Ceramics Printing with Strictly-Continuous Deposition. *ACM Transactions on Graphics* (Nov. 2019). https://doi.org/10.1145/3355089.3356509

Alexandra Ion, Robert Kovacs, Oliver S. Schneider, Pedro Lopes, and Patrick Baudisch. 2018. Metamaterial Textures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 336, 12 pages. https://doi.org/10.1145/3173574.3173910

Tim Kuipers, Willemijn Elkhuizen, Jouke Verlinden, and Eugeni Doubrovski. 2018. Hatching for 3D prints: Line-based halftoning for dual extrusion fused deposition modeling. *Computers & Graphics* 74 (aug 2018), 23–32. https://doi.org/10.1016/j.cag.2018.04.006

Gierad Laput, Xiang 'Anthony' Chen, and Chris Harrison. 2015. 3D Printed Hair: Fused Deposition Modeling of Soft Strands, Fibers, and Bristles. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. ACM, New York, NY, USA, 593–597. https://doi.org/10.1145/2807442.2807484

Xiaokang Liu, Lin Lu, Andrei Sharf, Xin Yan, Dani Lischinski, and Changhe Tu. 2020. Fabricable dihedral Escher tessellations. *Computer-Aided Design* 127 (2020), 102853. https://doi.org/10.1016/j.cad.2020.102853

Jonàs Martínez, Jérémie Dumas, Sylvain Lefebvre, and Li-Yi Wei. 2015. Structure and appearance optimization for controllable shape design. *ACM Transactions on Graphics* 34, 6 (nov 2015), 1–11. https://doi.org/10.1145/2816795.2818101

Jifei Ou, Gershon Dublon, Chin-Yi Cheng, Felix Heibeck, Karl Willis, and Hiroshi Ishii. 2016. Cilllia: 3D Printed Micro-Pillar Structures for Surface Texture, Actuation and Sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. ACM, New York, NY, USA, 5753–5764. https://doi.org/10.1145/2858036.2858257

Giuseppe Patanè, Michela Spagnuolo, and Bianca Falcidieno. 2004. Para-Graph: Graph-Based Parameterization of Triangle Meshes with Arbitrary Genus. *Computer Graphics Forum* 23, 4 (2004), 783–797. https://doi.org/10.1111/j.1467-8659.2004.00808.x

Ruggero Pintus, Enrico Gobbetti, Paolo Cignoni, and Roberto Scopigno. 2010. Shape enhancement for rapid prototyping. *The Visual Computer* 26, 6 (2010), 831–840.

G. Reeb. 1946. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *C. R. Acad. Sci. Paris* 222 (1946), 847–849.

Tim Reiner, Nathan Carr, Radomír Měch, Ondřej Šťava, Carsten Dachsbacher, and Gavin Miller. 2014. Dual-color Mixing for Fused Deposition Modeling Printers. *Comput. Graph. Forum* 33, 2 (May 2014), 479–486. https://doi.org/10.1111/cgf.12319

Haichuan Song, Jonàs Martínez, Pierre Bedell, Noemie Vennin, and Sylvain Lefebvre. 2019. Colored fused filament fabrication. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–11.

Ryo Suzuki, Tom Yeh, Koji Yatani, and Mark D. Gross. 2017. Autocomplete Textures for 3D Printing. (Mar 2017). arXiv:1703.05700v1 [cs.HC]

Haruki Takahashi and Jeeeun Kim. 2019. 3D Printed Fabric: Techniques for Design and 3D Weaving Programmable Textiles. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 43–51. https://doi.org/10.1145/3332165.3347896

Haruki Takahashi and Homei Miyashita. 2016. Thickness Control Technique for Printing Tactile Sheets with Fused Deposition Modeling. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16 Adjunct)*. ACM, New York, NY, USA, 51–53. https://doi.org/10.1145/2984751.2985701

Haruki Takahashi and Homei Miyashita. 2017. Expressive Fused Deposition Modeling by Controlling Extruder Height and Extrusion Amount. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 5065–5074. https://doi.org/10.1145/3025453.3025933

Cesar Torres, Tim Campbell, Neil Kumar, and Eric Paulos. 2015. HapticPrint: Designing Feel Aesthetics for Digital Fabrication. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. ACM, New York, NY, USA, 583–591. https://doi.org/10.1145/2807442.2807492

Chelsea Tymms, Esther P. Gardner, and Denis Zorin. 2018. A Quantitative Perceptual Model for Tactile Roughness. *ACM Transactions on Graphics* 37, 5 (nov 2018), 1–14. https://doi.org/10.1145/3186267

Chelsea Tymms, Siqi Wang, and Denis Zorin. 2020. Appearance-Preserving Tactile Optimization. *ACM Trans. Graph.* 39, 6, Article 212 (Nov. 2020), 16 pages. https://doi.org/10.1145/3414685.3417857

Kiril Vidimče, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. 2013. OpenFab: A Programmable Pipeline for Multi-Material Fabrication. *ACM Trans. Graph.* 32, 4, Article 136 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461993

Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 93–117.

Mark Wheadon. 2021. Velocity-Painting 3D printing companion. https://www.velocitypainting.xyz/gui/index.html

Jingru Yang, Sha He, and Lin Lu. 2019. Binary Image Carving for 3D Printing. *Computer-Aided Design* 114 (2019), 191–201. https://doi.org/10.1016/j.cad.2019.05.028

Kentaro Yasu. 2017. Magnetic Plotter: A Macrotexture Design Method Using Magnetic Rubber Sheets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 4983–4993. https://doi.org/10.1145/3025453.3025702

Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2016. Designing Structurally-sound Ornamental Curve Networks. *ACM Trans. Graph.* 35, 4, Article 99 (July 2016), 10 pages. https://doi.org/10.1145/2897824.2925888

Caowei Zhang, Guanyun Wang, Ye Tao, Xuan Li, Xin Liu, Chuqi Tang, Cheng Yao, and Fangtian Ying. 2017. infoTexture: Incremental Interfaces on Mesh Prototyping. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI EA '17)*. ACM, New York, NY, USA, 2263–2268. https://doi.org/10.1145/3027063.3053124

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2005. Feature-Based Surface Parameterization and Texture Mapping. *ACM Trans. Graph.* 24, 1 (Jan. 2005), 1–27. https://doi.org/10.1145/1037957.1037958

Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2016. Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

Kun Zhou, Xin Huang, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. 2006. Mesh Quilting for Geometric Texture Synthesis. *ACM Trans. Graph.* 25, 3 (July 2006), 690–697. https://doi.org/10.1145/1141911.1141942