Vis Comput
DOI 10.1007/s00371-017-1396-3

ORIGINAL ARTICLE

Marbling-based creative modelling

Shufang Lu[1] · **Yue Huang**[2] · **Xiaogang Jin**[2] · **Aubrey Jaffer**[3] · **Craig S. Kaplan**[4] · **Xiaoyang Mao**[5]

Abstract Most mathematical marbling simulations generate patterns for texture mapping and surface decoration. We explore the application of three-dimensional deformations inspired by mathematical marbling as a suite of tools to enable creative shape design. Our tools are expressed as analytical functions of space and are volume-preserving vector fields, meaning that the modelling process preserves volumes and avoids self-intersections. Complicated deformations are easily combined to create complex objects from simple ones. To achieve smooth and high-quality shapes, we also present a mesh refinement and simplification algorithm adapted to our deformations. We show a number of examples of shapes created with our technique in order to demonstrate its power and expressiveness.

Keywords Computational geometry · Shape modelling · Volume-preserving vector fields · Marbling art

1 Introduction

Three-dimensional models are a cornerstone of computer graphics practice, and research on modelling can be found throughout computer graphics, computer-aided design and computer-aided geometric design. Traditionally, such research has focused on fundamental questions of validity and robustness, but more recent work has also embraced other goals, such as aesthetics, functionality and creativity [9].

Marbling is a traditional craft in which a complex and attractive design is transferred to paper and fabrics [21]. Figure 1 shows two examples of 2D marbling patterns. The design is created by dropping multiple colours of paint onto water and then stirring the surface with a variety of tools such as brushes and combs. The resulting pattern of paint can then be applied to a sheet of paper or fabric by laying it atop the water surface. These vivid, unique patterns are used in many decorative contexts, including books, tissue boxes, jewellery and scarves. Past research in computer graphics has sought to reproduce the appearance of marbling (see Sect. 2.3). Typically, the thickness of the floating paint is considered negligible, reducing the marbling problem to a 2D physical simulation. Of course, real-world marbling may involve complex 3D fluid flow, and we believe that a fuller simulation of the 3D behaviour of marbling may open up new opportunities for creative 3D modelling.

Most past approaches to computer marbling have been based on traditional fluid flow simulation, i.e. advection in a discretized grid. However, a few researchers have explored methods based on explicit surface tracking, producing the geometric curves that define the boundaries between paints of different colours [4,19]. Inspired by these latter methods, we propose a simple and effective mathematical approach to 3D marbling. We define marbling tools in terms of volume-preserving 3D vector fields, which are then applied to the current geometry of a marbled shape in order to deform it. When a tool is moved, it causes a deformation of the shape along the tool's path. More complicated deformations can be achieved by applying several tools in sequence.

Electronic supplementary material The online version of this article (doi:10.1007/s00371-017-1396-3) contains supplementary material, which is available to authorized users.

✉ Xiaogang Jin
jin@cad.zju.edu.cn

1 Zhejiang University of Technology, Hangzhou, China

2 Zhejiang University, Hangzhou, China

3 Digilant, Boston, MA, US

4 University of Waterloo, Waterloo, ON, Canada

5 University of Yamanashi, Kofu, Yamanashi, Japan

Published online: 06 May 2017

Fig. 1 2D marbling patterns

The main contributions of this paper can be summarized as follows:

1. We introduce marbling as a novel tool in 3D geometric modelling. To the best of our knowledge, this is the first use of marbling as a technique for deforming 3D surfaces.
2. We develop volume-preserving homeomorphisms pattern functions. The 3D modelling process therefore preserves volumes and avoids self-intersections.
3. We improve on previous mathematical marbling methods by adding a new "welding" tool (Sect. 3.7), and propose alternative decay modes in other tools.
4. We offer an adaptive remeshing framework tailored to the deformations introduced in marbling, in order to overcome meshing artefacts that accumulate during deformation (Sect. 4).

2 Related work

In this section, we focus on techniques targeting deformation-based shape modelling, data-driven object modelling and the generation of marbling patterns, the topics that are most relevant to our work.

2.1 Deformation-based shape modelling

Barr [6] first introduced techniques for spatial deformation, defining transformation functions for tapering, twisting and bending. A deformation of a smooth surface was calculated using transformation matrices that varied smoothly along one coordinate axis. Blanc [7] proposed a generic implementation of Barr's method by extending deformations to functions of several spatial coordinates. These methods are easy to implement and involve little user interaction, but are limited to the three previously mentioned deformation types.

Free-form deformation is a well-established deformation technique in geometric modelling. It has been widely used in both academia and industry [26]. FFD based on Bézier basis functions was originally introduced by Sederberg and Parry [25]. Hsu et al. [14] introduced an intuitive direct manipulation approach for FFD, in which users move points on the surface being edited instead of the control points of a cage around the surface. Most recently, Cui et al. [10] proposed a real-time, GPU-based FFD method that can produce effective deformations of smooth regions of surfaces while also preserving sharp features.

Angelidis et al. [5] proposed a space deformation framework called Sweepers for interactive shape modelling. The space deformation operations are defined via paths through which a tool is swept. Each vertex in a surface is transformed by an amount determined by a scalar distance field relative to the surface of the tool. A related method is warp sculpting, introduced by Gain and Marais [12]. Their sculpting tools are also encoded by distance fields and decay functions. Kil et al. [18] introduced the 3D warp brush for interactive modelling. It uses the same principles, but the work concentrates more on user interaction. They adopted an immersive virtual reality environment which enables creative and intuitive direct manipulation of shapes. The work of von Funck et al. [28,29] applied scalar and vector fields for space deformation in a more direct way. The transformation of each vertex is obtained via path integration of a time-based vector field. Since their vector fields are divergence-free, the resulting deformations were volume preserving and free of self-intersections.

Other related deformation-based shape modelling methods include deformations with nonlinear bijective mappings [23], user-controlled nonlinear mesh deformations [22, 27] and shape-driven deformations [24].

2.2 Data-driven object modelling

Data-driven modelling methods produce novel shapes by assembling parts extracted from 3D model databases. Such techniques are playing an increasingly important role in geometric modelling. Inspired by evolutionary computation, Xu et al. [32] proposed a genetic approach for 3D shape generation. They define crossover and mutation operators to perform part replacement and warping on an initial generation of shapes. The evolution process is guided by a fitness function derived from user feedback. Guo et al. [13] introduced a system based on shape grammars to generate creative 3D monster models to serve as references for artists. Based on a large collection of pre-segmented 3D models, Xie et al. [30] developed an interactive sketch-to-design system that generates novel models by combining parts from the collection. Huang et al. [15] used deep learning to synthesize geometrically diverse 3D shape families. On a more philosophical note, Cohen-Or and Zhang [9] discussed object modelling in the context of computational creativity, articulating a sequence of paradigms for tools to assist users with the construction of creative models.

2.3 Marbling patterns

We divide the methods for synthesis of marbling patterns into two categories: texture-based and shape-based simulation.

Texture-based simulation Most marbling simulation methods generate 2D patterns for texture mapping and surface decoration. Akgun [3] introduced a digital marbling tool for generating traditional Turkish marbling textures. Acar [2] developed a level-set method to model various flows and effects in 2D fluid simulation, including marbling patterns. Jin et al. [17] proposed an efficient GPU-based marbling system. Xu et al. [31] and Zhao et al. [33] improved the method of Jin et al. by using a non-dissipative fluid solver, suppressing blurring artefacts. These methods all produce 2D textures, which will inevitably undergo deformation when mapped non-isometrically onto 3D surfaces. Lu et al. [20] first proposed a marbling method that produces 3D solid textures, which can then be applied effortlessly to a wide range of 3D objects without any distortion.

Shape-based simulation Shape-based marbling simulation methods produce geometric descriptions of 2D marbling-like shapes, i.e. they generate vector graphics rather than raster images. Ando and Tsuruno [4] adopted an explicit surface tracking method to create 2D vector graphic marbling shapes. They used sequences of connected points to describe contours between paints and then advected or stretched them along an underlying velocity field generated by fluid simulation. In addition to marbling, this method has applications in shape design, target-driven shape deformation, vector graphic editing and Flash animation. However, their underlying fluid simulation required significant computation. Lu et al. [19] introduced an algorithm that expressed standard marbling tools as closed-form mathematical expressions, yielding marbling drawings as vector graphic images. However, these methods are still restricted to two dimensions.

Our work can be viewed as a hybrid of the approaches mentioned above: a 3D deformation model inspired by marbling simulation, but one that tracks deforming surfaces rather than advecting fluid in a grid. Furthermore, our target application is creative surface design, not texture generation.

3 Deformation patterning tools

In our work, every tool is defined formally as a continuous deformation of 3D space. Let $f : \mathbb{R}^3 \mapsto \mathbb{R}^3$ be a continuous, one-to-one function that maps points to points in 3D. A given surface S may be deformed into a new surface S' by defining $S' = \{f(\boldsymbol{P})|\boldsymbol{P} \in S\}$. More generally, if $\{f_i\}_{i=1}^n$ is a sequence of homeomorphisms, then we can define $S_0 = S$ and $S_i = \{f_i(\boldsymbol{P})|\boldsymbol{P} \in S_{i-1}\}$ for $i = 1, \ldots, n$. The final surface S_n will be the result of applying each of the n deformation tools in sequence. Because each f_i is continuous and one to one, it is guaranteed never to tear the surface into multiple pieces and will never cause any two parts of the surface to collide. Any composition of such functions satisfies these same properties, in particular the composite deformation

$$S_n = \{(\Omega_{i=1}^n f_i)(\boldsymbol{P})|\boldsymbol{P} \in S\}, \tag{1}$$

where $\Omega_{i=1}^n f_i$ is shorthand for the iterated composition $f_n \circ \cdots \circ f_1$. Note that the ordering of these operations is important. In particular, the final deformation f_n will have the most visible effect in S_n.

In the following subsections, we define seven patterning tools by giving closed-form expressions for their deformation functions. The functions are taken from, and retain the behaviours typical of, existing 2D and solid mathematical marbling techniques [19,20]. They are all volume preserving (see "Appendix 1" for more detailed discussion of this property). In addition, we introduce a new welding tool (Sect. 3.7) and provide an alternative decay mode that produces output more closely resembling fluid flow simulation ("Appendix 2").

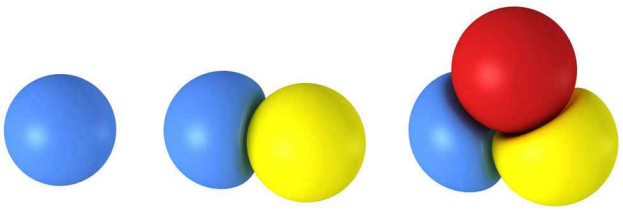


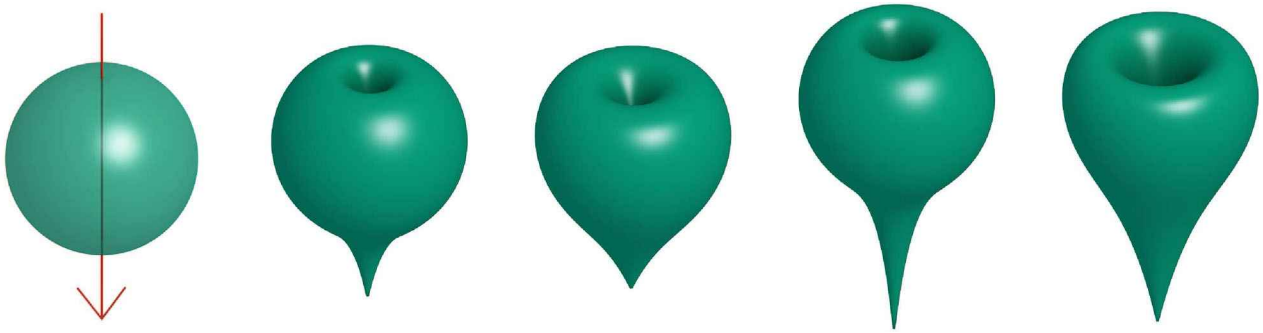
Fig. 2 The process of placing sphere objects to 3D space by the sphere insertion pattern function. The previously dropped objects are deformed by the subsequently injected ones

3.1 Sphere insertion function

Given a sphere with centre $\boldsymbol{C}$ and radius r, we wish to model the effect of inserting the sphere into an existing environment, by developing a deformation that pushes 3D geometry out of the way to make room for the incoming sphere. We use the following function:

$$f(\boldsymbol{P}) = \boldsymbol{C} + (\boldsymbol{P} - \boldsymbol{C})\sqrt[3]{1 + \frac{r^3}{|\boldsymbol{P} - \boldsymbol{C}|^3}}. \tag{2}$$

Figure 2 shows the results of injecting spheres to 3D space by using the sphere insertion pattern function. This function is defined everywhere in $\mathbb{R}^3$ except for C; in practice, it is unlikely that a pre-existing mesh vertex would lie precisely at C; if one is found, it can be jittered slightly away from C before deformation.



(a) **(b)** $\alpha = 5, \lambda = 0.3$ **(c)** $\alpha = 5, \lambda = 0.6$ **(d)** $\alpha = 10, \lambda = 0.3$ **(e)** $\alpha = 10, \lambda = 0.6$

Fig. 3 Line patterns. The input sphere and line are shown (**a**), followed by the effect of the deformation for various combinations of α and λ (**b**–**e**)

3.2 Line pattern function

Let a 3D line be defined by a point $\boldsymbol{A}$ on the line and a unit vector $\mathbf{m}$ that points in the direction of the line. For any given point $\boldsymbol{P}$, let $d(\boldsymbol{P})$ denote the minimum distance from $\boldsymbol{P}$ to the line. We can then define the deformation function

$$f(\boldsymbol{P}) = \boldsymbol{P} + \alpha\lambda^{d(\boldsymbol{P})}\mathbf{m}. \tag{3}$$

This function displaces a vertex $\boldsymbol{P}$ in the direction $\mathbf{m}$ by an amount that decreases exponentially with d. It is controlled by two constants α and λ, with the constraint $\lambda < 1$. As α increases, the central axis of the deformation (i.e. the line itself) is displaced by a greater amount. As λ decreases, the bend in the deformation becomes sharper. Figure 3b–e shows the results of deforming along the same line for different values of α and λ. Previous mathematical marbling techniques [19,20] used hyperbolic approximations to model displacement relative to a line. In "Appendix 2", we explain why the exponential approximation can better simulate the fluid-like effects of marbling.

3.3 Comb pattern function

In marbling, a comb tool is a set of evenly spaced parallel lines that move through the paint layer in tandem. While it might be possible to simulate a line set by composing multiple instances of Eq. 3, it is much more efficient to define a modified distance function that encapsulates the periodicity of the lines, simulating an infinite sequence in a single step. Accordingly, we apply a modified version of Eq. 3 in which the distance function $d(\boldsymbol{P})$ has been replaced by the function

$$d_{\text{comb}}(\boldsymbol{P}) = k/2 - |\text{fmod}(d(\boldsymbol{P}), k) - k/2|. \tag{4}$$

Here, k is used to control the spacing between the lines, and the base function $d(\boldsymbol{P})$ is used to measure the distance to an arbitrary reference line defined as in Sect. 3.2.

3.4 Spherical and cylindrical shell patterns

In addition to displacing space in the direction of a line, we can swirl space around that line. For any vector $\mathbf{v}$, let $R_{\mathbf{v}}(\theta)$ represent a rotation by angle θ about $\mathbf{v}$. The matrix form of this rotation can be computed using the standard axis-angle representation [16, Sect. 11.2.3]. Let $\boldsymbol{A}$ and $\mathbf{m}$ define a 3D line, as before. We define two related tools that rotate about this line, where the rotation is maximized for points that lie on a spherical or cylindrical shell.

To rotate relative to a cylindrical shell, we imagine a cylinder of radius r centred on the axis defined by $\boldsymbol{A}$ and $\mathbf{m}$. Once again, let $d(\boldsymbol{P})$ denote the minimum distance from $\boldsymbol{P}$ to this axis, from which $d_{\text{shell}}(\boldsymbol{P}) = |d(\boldsymbol{P}) - r|$ is the unsigned distance to the surface of the cylinder. We can now define the deformation function

$$f(\boldsymbol{P}) = \boldsymbol{A} + R_{\mathbf{m}}(\alpha\lambda^{d_{\text{shell}}(\boldsymbol{P})}/d(\boldsymbol{P}))(\boldsymbol{P} - \boldsymbol{A}), \tag{5}$$

where α and λ control the amount of rotation in a similar manner to Sect. 3.2. We can also define a tool with maximum rotation on a spherical shell centred at $\boldsymbol{A}$ simply by replacing the distance function above with $d(\boldsymbol{P}) = ||\boldsymbol{P} - \boldsymbol{A}||$. The effect of the spherical shell tool is visualized in Fig. 4b.

3.5 Spherical and cylindrical vortex patterns

The shell functions of the preceding section measure unsigned distance to a cylinder axis or sphere centre, causing the swirling direction to reverse abruptly on the surface of the shell. That behaviour is reasonable as a simulation of a knife cutting through paint along a circular trajectory.

If we instead use a signed distance function, that is, if we substitute $d_{\text{vortex}}(\boldsymbol{P}) = d(\boldsymbol{P}) - r$ for d_{shell} in Eq. 5, we obtain a vortex tool in which the swirling continues smoothly across the shell. This style of deformation is a valuable addition in the context of creative modelling. Figure 4c illustrates the effect of the spherical vortex tool.

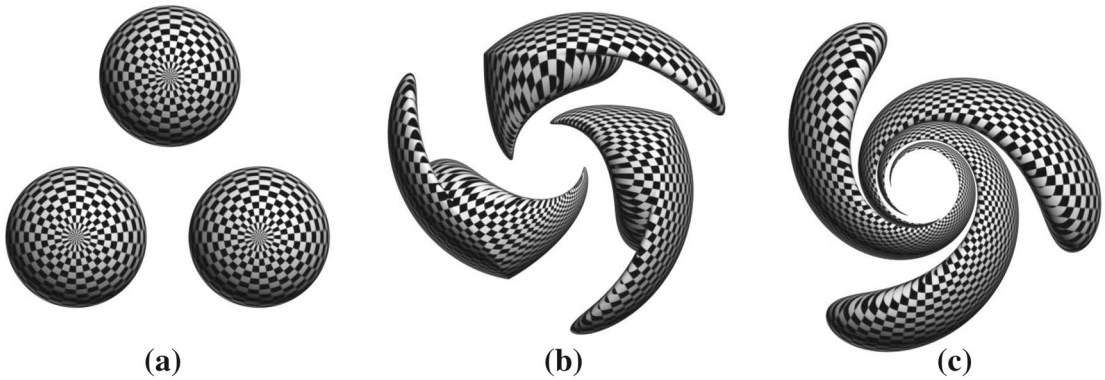


Fig. 4 An initial environment consisting of three spheres (**a**), together with the results of applying a spherical shell tool (**b**) and a spherical vortex tool (**c**) to the environment

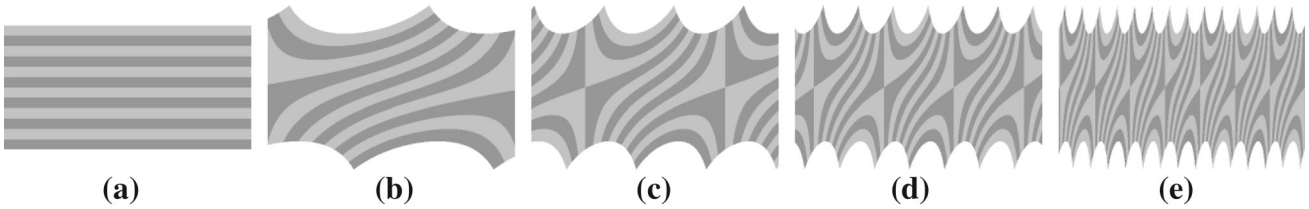


Fig. 5 2D pattern-welded bars with 0, 1/2, 1, 2, and 4 full twists (**a**–**e**) respectively

3.6 Wave displacement pattern function

Let $g(x, y)$ be a function that represents displacement in the z direction of points in the xy plane. There are several ways to define functions of this sort that produce waves in space. For example,

$$g(x, y) = a \sin(\omega_1 x + \phi_1) + b \sin(\omega_2 y + \phi_2) \tag{6}$$

defines a pattern of bumps resembling an egg carton, and

$$g(x, y) = a \sin\left(\omega\sqrt{x^2 + y^2} + \phi_1\right) \tag{7}$$

defines a pattern of evenly spaced concentric rings. We can now define a deformation function that offsets points using g. For any point $\boldsymbol{P} = (x, y, z)$, we define

$$f(\boldsymbol{P}) = f(x, y, z) = (x, y, z + g(x, y)). \tag{8}$$

If R represents any orientation in space via a rotation matrix, then the composition $R \cdot f \cdot R^{-1}$ applies waves in the orientation given by R instead of the z axis. Figure 9c shows the result of applying a wave pattern function after a comb pattern function.

3.7 Welding pattern function

In pattern welding, different metals are welded together by heating and hammering, and the resulting composite is bent and twisted (not unlike marbling) so that a pattern is formed by the alternation of the metals. The patterns create attractive surfaces for knife blades and swords, perhaps most famously in the form of Damascus steel.

Inspired by pattern welding, we introduce a helical twisting deformation as a new 3D marbling operation. As always, let the axis of the helix be represented by a point $\boldsymbol{A}$ and a direction $\mathbf{m}$. Let $\pi(\boldsymbol{P})$ denote the orthogonal projection of $\boldsymbol{P}$ onto the line defined by $\boldsymbol{A}$ and $\mathbf{m}$. Relying again on the axis-angle rotation $R_{\mathbf{m}}(\theta)$, we define

$$f(\boldsymbol{P}) = \boldsymbol{A} + R_{\mathbf{m}}(c||\pi(\boldsymbol{P}) - \boldsymbol{A}||)(\boldsymbol{P} - \boldsymbol{A}). \tag{9}$$

Here, c is the twist parameter, and larger c values twist tightly. Figure 5 visualizes the effect of pattern welding by showing 2D slices through simulated pattern-welded bars with 0, 1/2, 1, 2 and 4 full twists. The results are generated by twisting the XZ plane ([−250, 125, 250], [250, 125, 250]) along the axis by setting $A = (0, 0, 0)$, and $m = (0, 0, 1)$. Figure 9b is the 3D result by applying welding pattern function on an initial sphere.

4 Adaptive remeshing

The deformation patterning tools described in Sect. 3 are essentially volume preserving and intersection-free. Because they are continuous and one to one, continuous surfaces will be mapped to continuous surfaces. However, when surfaces

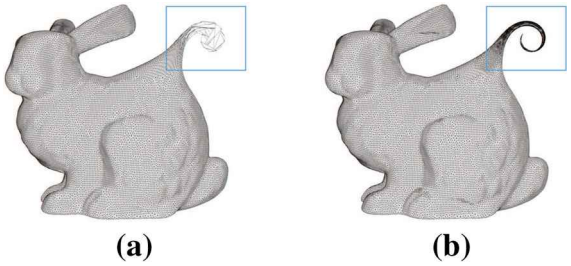


Fig. 6 Remeshing. **a** A deformed mesh with severe artefacts around the newly created geometry, **b** the final surface after remeshing

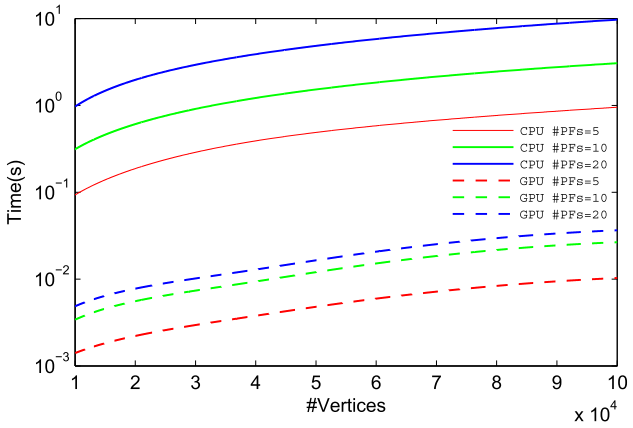


Fig. 7 The performance comparison of our method with CPU and GPU implementation

are sampled discretely via triangle meshes, the mesh vertices may not sample the underlying smooth surface densely enough to capture the details of the new shape that emerges after applying a deformation (Fig. 6a). The curved results associated with the shell and vortex tools may suffer from especially bad undersampling artefacts. We address this problem using an adaptive remeshing algorithm, inspired by the work of Brochu and Bridson [8].

We represent a triangle mesh M as a set of n vertices $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n, m$ edges $e_1, \ldots, e_m$, where each edge is an ordered pair of integers in $1 \ldots n$ representing a choice of two vertices, and a set of triangular faces $t_1, \ldots, t_k$, each one an ordered triple of integers in $1 \ldots n$. We can naturally apply a deformation f (which could be a composition of multiple tool applications) to a mesh by constructing a new mesh M' with vertices $\boldsymbol{P}'_i = f(\boldsymbol{P}_i)$ for all i and the same edges and faces as M. We must now process M' to account for any artefacts introduced during deformation. We use the following steps:

1. Estimate a normal vector $\mathbf{n}_i$ for all vertices $\boldsymbol{P}'_i$ in M', by averaging the face normals at faces of M' incident on $\boldsymbol{P}'_i$.
2. Consider every edge $e_j = (a_j, b_j)$ of M' in increasing order by length. If either the length of the edge exceeds a threshold $d_{\max}$ or the angle between the vertex normals exceeds an angle threshold $\theta_{\max}$, we refine the edge. Specifically, we compute the midpoint $\boldsymbol{P} = (\boldsymbol{P}_{a_j} + \boldsymbol{P}_{b_j})/2$ of the corresponding edge of M. We then add $\boldsymbol{P}$ to M and $\boldsymbol{P}' = f(\boldsymbol{P})$ to M' and update the edges and faces of both meshes to take the new vertex and its deformed counterpoint into account.
3. The preceding step may eventually produce meshes sampled more finely than necessary, so we also include a simplification step. As we are considering the edges of M' above, if either the length of the edge falls below a threshold $d_{\min}$ or the angle between its vertex normals falls below a threshold $\theta_{\min}$, we collapse the edge onto its midpoint. Simultaneously, we perform the same collapse on the original mesh to maintain the correspondence and update the edges and faces of both meshes to take the collapse into account.
4. Repeat the loop over all edges in Steps 2 and 3 until no further splits or collapses occur.

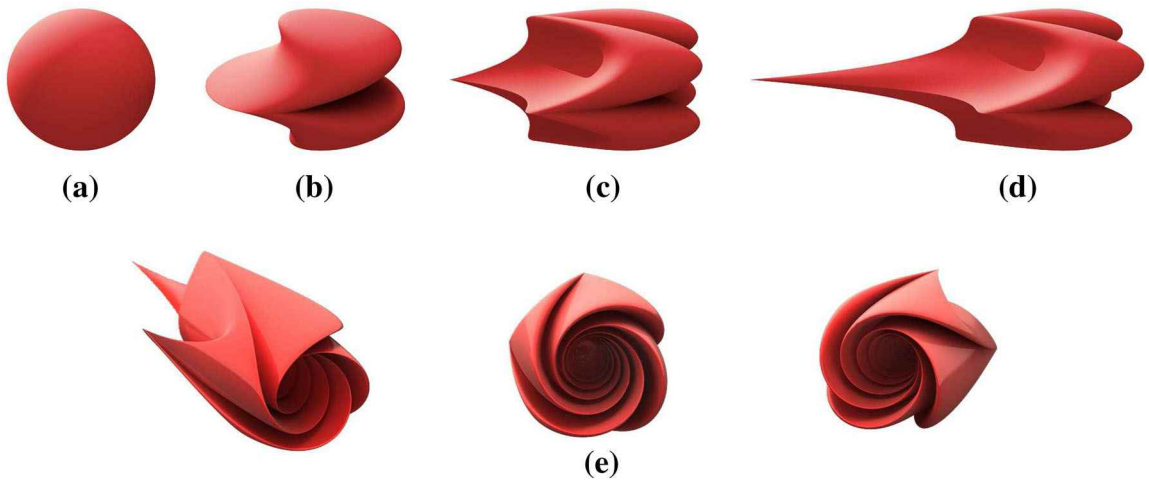


Fig. 8 A step-by-step demonstration of the construction of a flower shape using marbling tools

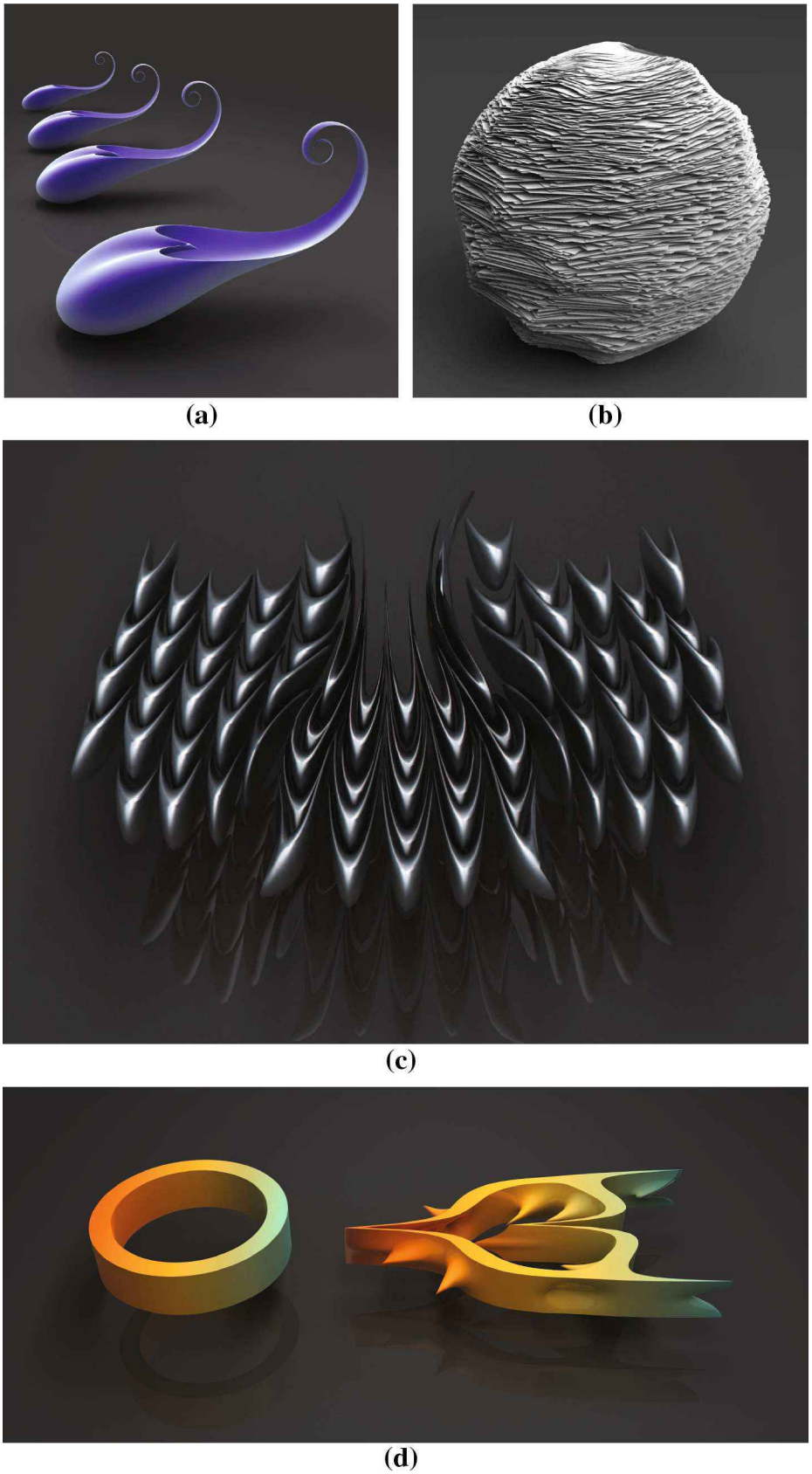


Fig. 9 Results of our marbling-based creative shape design. These complex objects are all created from simple ones

The angle between two vertices $\boldsymbol{P}'_a$ and $\boldsymbol{P}'_b$ is given by $\cos^{-1}(\mathbf{n}_a \cdot \mathbf{n}_b)$, where $\mathbf{n}_a$ and $\mathbf{n}_b$ are the vertex normals estimated above. We also compute the lengths of edges in a way that takes into account estimated curvature at vertices. If c_a and c_b are the estimated curvatures at $\boldsymbol{P}'_a$ and $\boldsymbol{P}'_b$, then we estimate the length of an edge joining these vertices as $||\boldsymbol{P}'_a - \boldsymbol{P}'_b||(c_a + c_b)/2$.

Figure 6 shows the effect of remeshing on a deformed bunny model.

5 Results and discussion

The running time of our marbling process depends on the number of mesh vertices used to represent all the geometry in the environment and the number of tools applied in the modelling session. Fortunately, much of the mesh processing is highly parallel and can take advantage of GPU acceleration. We store initial mesh vertices in individual kernels in CUDA, to be deformed by pattern functions.

To test the performance of our system, we ran it on a 3.2-GHz Intel Core i5-3470 CPU, with an NVIDIA GeForce GTX 660 GPU. Figure 7 shows the performance of our system with different mesh sizes and numbers of pattern functions. In the graph, the horizontal axis refers to the number of mesh vertices used, and the different curves correspond to different numbers of deformations applied. The solid lines give times for our CPU implementation, and the lower dashed lines are the corresponding GPU-based times. Clearly, hardware acceleration provides a speed boost of about two orders of magnitude.

Figure 8 shows an example of generating a rose shape starting with a simple sphere. Two sweeps of a comb tool in (b) and (c), in perpendicular directions, produce the four lobes on the right-hand side of the sphere. In (d), the line tool draws out a single sharp branch on the left. Finally, the vortex tool swirls the lobes together, producing the form shown from three different angles in (e).

Past research in deformation-based modelling has produced tools that deform 3D shapes in a volume-preserving, collision-free way. Typically, they define weight functions to control the amount of deformation on a surface. However, these past approaches tend to be computationally expensive. For example, vector field-based shape deformation [28] can be used interactive only on small numbers of samples [11]. As shown in Fig. 7, our method is highly efficient.

Compared to previous mathematical marbling methods [19,20] which are used for 2D and solid texture generation, we improve them to enable creative shape design in this paper. We enlarge the expressive vocabulary of tools with a new welding deformation (Sect. 3.7), as shown in Figs. 5 and 9b. In addition, we offer an alternative exponential decay mode instead of the earlier hyperbolic one. It is proven to be a closer approximation of the fluid flow simulation that underlies mathematical marbling (see "Appendix 2") and is able to achieve smooth deformation results. Figure 9 shows some representative results of our method. In (a–c), tools are applied to an initial sphere or spheres. The deformed shape in (d) is the result of applying deformations to the annulus shown on the left. Figure 9b is the result without remeshing.

6 Conclusions

The techniques presented in this paper can generate impressive 3D shapes through the application of tools inspired by mathematical marbling. The results are obtained by transforming the original shapes through explicit surface tracking. To achieve high-quality results, the deformation functions are designed to be volume preserving. Moreover, the deformed shapes are refined through an adaptive remeshing algorithm. The performance is also improved by porting the deformation algorithms to the GPU with CUDA, boosting speed by two orders of magnitude compared to a CPU implementation.

Our method has limitations. Our approach can generate meshes with fine details. The mesh refinement operation is done after all the pattern functions are applied onto the original mesh. Thus, users may cannot expect how the final shape looks like especially when severe artefacts occur in the design process. There are several avenues for future research. First, we will apply the method to other mesh representations since it does not rely on the connectivity information in a mesh. Second, we plan to develop more deformation patterning tools to achieve a wider variety of marbling-like 3D shapes.

Acknowledgements Shufang Lu was supported by the National Natural Science Foundation of China (Grant No. 61402410) and the China Scholarship Council. Xiaogang Jin was supported by the National Natural Science Foundation of China (Grant No. 61472351).

Appendix 1: Volume-preserving deformations

In this section, we argue that the tools we define lead to volume-preserving deformations of space. We adopt seven transform functions. Of these, the line, comb, shell, vortex, wave and welding patterns are homeomorphisms (i.e. continuous bijections) of $\mathbb{R}^3$, but the sphere insertion tool is not. For a continuously differentiable vector field $\boldsymbol{u}$, the volume form is preserved under the flow of a solenoidal vector field $\nabla \cdot \boldsymbol{u} = 0$ [1].

We can regard our tools as vector fields by considering the vector by which they displace every point in space. That is, we define a vector field $\boldsymbol{u}(x, y, z) = f(\boldsymbol{P}) - \boldsymbol{P}$. For each of the first five tools mentioned above, it is easy to prove that its divergence equals zero.

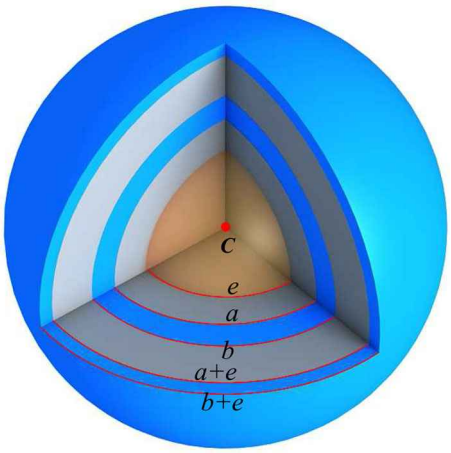


Fig. 10 Sphere insertion tool satisfies the volume-preserving property

The divergence of the welding function is not well defined as x, y or z goes to infinity. So we cannot use the divergence test for the welding function. But it is volume preserving for bounded x, y, z. Slice 3-space into infinitesimally thin slices perpendicular to the twist axis. Each slice is then simply rotated around the twist axis. Because the rotation does not change the areas in each slice, the volumes are preserved.

The sphere insertion function is not continuously differentiable around the injection point $\boldsymbol{C}$, so its divergence is not well defined. However, we can show that this transform preserves the volume of all neighbourhoods not containing $\boldsymbol{C}$. As shown in Fig. 10, consider the neighbourhood having solid angle Ω of the spherical shell centred on $\boldsymbol{C}$ having inner radius a and outer radius b. Its volume is $(4/3)\pi\,\Omega\,(b^3 - a^3)$. If a sphere of radius e is injected at $\boldsymbol{C}$, the new spherical region centred on $\boldsymbol{C}$ will have volume $(4/3)\pi e^3$ and increase the radius from $\boldsymbol{C}$ of all other points. The radially symmetric expansion does not change the solid angle Ω. The radius of the outer shell increases from b to $\sqrt[3]{b^3 + e^3}$; the inner shell radius increases from a to $\sqrt[3]{a^3 + e^3}$. Because $(b^3 + e^3) - (a^3 + e^3) = b^3 - a^3$, the volume of this neighbourhood remains $(4/3)\pi\,\Omega\,(b^3 - a^3)$. Neighbourhoods with other shapes can be assembled from these shell fragments, each preserving its volume under injection, so long as each fragment does not include $\boldsymbol{C}$. Therefore, the sphere insertion function is volume preserving at all locations except the point of injection, $\boldsymbol{C}$.

Appendix 2: Exponential decay mode

Consider an unbounded plane containing a two-dimensional incompressible laminar flow. Given x, y as the coordinates of the point $\boldsymbol{P}$, associated with $f(x, y)$ is a vector field $\boldsymbol{H}(x, y) = \boldsymbol{F}(x, y) - (x, y)$ returning the vector displacement at each coordinate. Along the y axis, we introduce a displacement α, $\boldsymbol{H}(0, y) = (0, \alpha)$ $[\boldsymbol{F}(0, y) = (0, y + \alpha)]$. This displacement will not affect points far away from the x axis; so the limit of $\boldsymbol{H}(x, y)$ tends to zero as the magnitude of x grows. Because the fluid is incompressible, the divergence of $\boldsymbol{H}$ is zero everywhere. Because its flow is laminar, it is uniform in the direction of motion, y: $\frac{\partial \boldsymbol{H}_y}{\partial y} = 0$.

Thus, $\boldsymbol{H}(x, y)$ depends only on x. Furthermore, only the y component of $\boldsymbol{H}(x, y)$ depends on x. Let $f(x) = \boldsymbol{H}_y(x, 0)$; then, $f(0) = \alpha$. $f(x)$ is even; the displacements to either side of $x = 0$ will be equal and in the same direction. So we will consider $f(x)$ for $x \geq 0$ only. Because the flow is laminar, displacements induced by α travel along the x axis should be proportional to α. Let $A = f(b)$. Then, $f(2b)$ will be reduced from A by the same proportion as A was reduced from α: $f(2b) = \frac{A^2}{\alpha}$. Thus, $\frac{f(2b)}{f(0)} = \frac{A^2}{\alpha^2} = \frac{f(b)^2}{f(0)^2}$.

The only continuous real functions satisfying these constraints are $f(x) = \alpha\lambda^{|x|}$ with independent parameter $0 < \lambda < 1$ related to the viscosity.

References

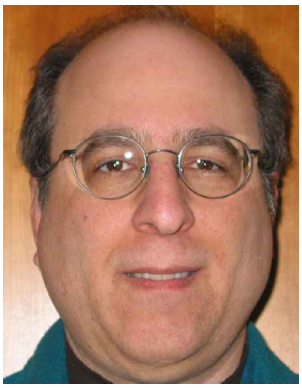
1. https://en.wikipedia.org/wiki/Volume_form
2. Acar, R.: Level set driven flows. ACM Trans. Graph. (TOG) **26**(4), 15 (2007)
3. Akgun, B.T.: The digital art of marbled paper. Leonardo **37**(1), 49–52 (2004)
4. Ando, R., Tsuruno, R.: Vector graphics depicting marbling flow. Comput. Graph. **35**(1), 148–159 (2011)
5. Angelidis, A., Wyvill, G., Cani, M.P.: Sweepers: swept deformation defined by gesture. Graph. Model **68**(1), 2–14 (2006)
6. Barr, A.H.: Global and local deformations of solid primitives. ACM Siggraph Comput. Graph. **18**(3), 21–30 (1984)
7. Blanc, C.: Generic implementation of axial deformation techniques. Graph. Gems **5**, 249–256 (1995)
8. Brochu, T., Bridson, R.: Robust topological operations for dynamic explicit surfaces. SIAM J. Sci. Comput. **31**(4), 2472–2493 (2009)
9. Cohen-Or, D., Zhang, H.: From inspired modeling to creative modeling. Vis. Comput. **32**(1), 7–14 (2016)
10. Cui, Y., Feng, J.: GPU-based smooth free-form deformation with sharp feature awareness. Comput. Aided Geom. Des. **35**, 69–81 (2015)
11. Gain, J., Bechmann, D.: A survey of spatial deformation from a user-centered perspective. ACM Trans. Graph. (TOG) **27**(4), 107 (2008)
12. Gain, J., Marais, P.: Warp sculpting. IEEE Trans. Vis. Comput. Graph. **11**(2), 217–227 (2005)
13. Guo, X., Lin, J., Xu, K., Jin, X.: Creature grammar for creative modeling of 3d monsters. Graph. Models **76**(5), 376–389 (2014)
14. Hsu, W.M., Hughes, J.F., Kaufman, H.: Direct manipulation of free-form deformations. ACM Siggraph Comput. Graph. **26**(2), 177–184 (1992)
15. Huang, H., Kalogerakis, E., Marlin, B.: Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces. Comput. Graph. Forum **34**(5), 25–38 (2015)
16. Hughes, J.F., Van Dam, A., McGuire, M., Sklar, D., Foley, J.D., Feiner, S.K., Akeley, K.: Computer Graphics: Principles and Practice, 3rd edn. Addison Wesley, Boston (2014)

17. Jin, X., Chen, S., Mao, X.: Computer-generated marbling textures: a GPU-based design system. IEEE Comput. Graph. Appl. **27**(2), 78–84 (2007)
18. Kil, Y.J., Renzulli, P., Kreylos, O., Hamann, B., Monno, G., Staadt, O.G.: 3d warp brush modeling. Comput. Graph. **30**(4), 610–618 (2006)
19. Lu, S., Jaffer, A., Jin, X., Zhao, H., Mao, X.: Mathematical marbling. IEEE Comput. Graph. Appl. **32**(6), 26–35 (2012)
20. Lu, S., Jin, X., Jaffer, A., Gao, F., Mao, X.: Solid mathematical marbling. IEEE Comput. Graph. Appl. **37**(2), 90–98 (2017)
21. Maurer-Mathison, D.V.: The Ultimate Marbling Handbook: A Guide to Basic and Advanced Techniques for Marbling Paper and Fabric. Watson-Guptill, New York (1999)
22. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Fibermesh: designing freeform surfaces with 3d curves. ACM Trans. Graph. (TOG) **26**(3), 41 (2007)
23. Pentland, A., Williams, J.: Good vibrations: modal dynamics for graphics and animation. ACM Siggraph Comput. Graph. **23**(3), 207–214 (1989)
24. Schmitt, B., Pasko, A., Schlick, C.: Shape-driven deformations of functionally defined heterogeneous volumetric objects. In: GRAPHITE2003, ACM, New York, 127–134 2003
25. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. ACM Siggraph Comput. Graph. **20**(4), 151–160 (1986)
26. Sieger, D., Menzel, S., Botsch, M.: On shape deformation techniques for simulation-based design optimization. In: New Challenges in Grid Generation and Adaptivity for Scientific Computing, Springer, Berlin, 281–303 2015
27. Singh, K., Fiume, E.: Wires: a geometric deformation technique. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, ACM, New York, 405–414 1998
28. Von Funck, W., Theisel, H., Seidel, H.P.: Vector field based shape deformations. ACM Trans. Graph. (TOG) **25**(3), 1118–1125 (2006)
29. Von Funck, W., Theisel, H., Seidel, H.P.: Explicit control of vector field based shape deformations. In: 15th Pacific Conference on Computer Graphics and Applications, 2007. PG'07, pp. 291–300. IEEE, Piscataway (2007)
30. Xie, X., Xu, K., Mitra, N.J., Cohen-Or, D., Gong, W., Su, Q., Chen, B.: Sketch-to-design: context-based part assembly. Comput. Graph. Forum **32**(8), 233–245 (2013)
31. Xu, J., Mao, X., Jin, X.: Nondissipative marbling. IEEE Comput. Graph. Appl. **28**(2), 35–43 (2008)
32. Xu, K., Zhang, H., Cohen-Or, D., Chen, B.: Fit and diverse: set evolution for inspiring 3d shape galleries. ACM Trans. Graph. (TOG) **31**(4), 57 (2012)
33. Zhao, H., Jin, X., Lu, S., Mao, X., Shen, J.: Atelierm++: a fast and accurate marbling system. Multimed. Tool Appl. **44**(2), 187–203 (2009)

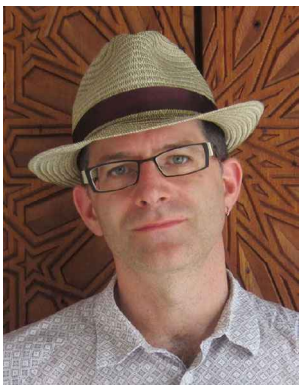
Shufang Lu is an assistant professor at College of Computer Science and Technology, Zhejiang University of Technology. She is currently a visiting scholar at the Computer Graphics Lab, the University of Waterloo. She received her B.Sc. degree in software engineering from Wuhan University, and M.Sc. and Ph.D. degrees in computer science and technology from Zhejiang University. Her research interests include non-photorealistic rendering and image processing.

Yue Huang is a postgraduate student of the State Key Lab of CAD&CG, Zhejiang University. She received her B.Sc. degree in digital media technology from Zhejiang University. Her research interests include virtual try-on, modeling and virtual reality.

Xiaogang Jin is a professor of the State Key Lab of CAD&CG, Zhejiang University. He received his B.Sc. degree in computer science in 1989, and M.Sc. and Ph.D. degrees in applied mathematics in 1992 and 1995, all from Zhejiang University. His current research interests include virtual try-on, insect swarm simulation, traffic simulation, implicit surface modeling and applications, creative modeling, sketch-based modeling and image processing. He received an ACM Recognition of Service Award in 2015. He is a member of the IEEE and ACM.



Aubrey Jaffer is a mathematician and data scientist at Digilant. His research interests include convection, radiative transfer, numerical analysis, algebraic geometry, symbolic algebra and space filling curves. Aubrey has a BS in mathematics from the Massachusetts Institute of Technology.



Craig S. Kaplan is an associate professor of Computer Science at the University of Waterloo. He has a BMath in pure mathematics and computer science from Waterloo, and an M.S. and Ph.D. in computer science from the University of Washington. Kaplan studies the application of computer graphics and mathematics to problems in art, architecture and design and is an expert on topics such as Islamic geometric patterns and computational applications of tiling theory. He also enjoys experimenting with computer-aided art and design using modern technology like 3D printing.

Xiaoyang Mao is a professor in the Department of Computer and Media Engineering at the University of Yamanashi, Japan. Her research interests include nonphotorealistic rendering, texture synthesis, perception and affect-based rendering, and visualization. Mao has a Ph.D. in computer science from the University of Tokyo.