Mathematical Marbling

Shufang Lu, Aubrey Jaffer, Xiaogang Jin, Hanli Zhao, and Xiaoyang Mao

Abstract—Marbling is the art of creating stone-like or intricate abstract decorations from liquid inks floating on water or gel. Although the fluid dynamics of marbling processes can be simulated, we introduce a mathematical approach with closed-form expressions. Our approach improves control, ease of implementation, parallelism, and speed, which enables real-time visual feedback as well as the creation of vivid flowing animations. Designs can be started from a blank sheet or raster images and videos. When started from a blank sheet, our approach can produce compact resolution-independent vector outputs. The transforms for our marbling operations all have inverse transforms. Forward application is used to generate compact resolution-independent vector-based output; inverse application is used to generate pixel-based output. In both cases, the closed-form expressions preserve the quality and sharpness of the designs. The efficiency and effectiveness of our method are demonstrated via extensive comparisons with existing digital marbling techniques. We also show various applications including cyclic texture tiling, scene decoration, surface detail rendering for 3D objects, image editing, and interactive video processing.

Index Terms—Marbling art, vector image, closed-form expression, surface details rendering.

1 Introduction

Marbling is a decorative art with several distinct traditions originating in Asia, perhaps as long as 1000 years ago. It spread to Europe in the 16th century where its primary application was producing endpapers and covers for books. Mechanized bookbinding caused the decline of marbling in the West; but it has enjoyed a revival as a folk art since the 1970s. Some of the Asian traditions have continuity from ancient times; in Japan, the Hiroba Family claims to have made marbled paper since 1151. Although primarily used for decoration, marbling has security applications. Marbling the edges of ledger books makes missing pages apparent and documents written over pale marbling are tamper-resistant.

The beauty of the swirling and wavy patterns, the use of different kinds of tools, and the fascination of color dispersion all contribute to its distinctive appeal. Historically, traditional designs are abstract or have a stone or fibrous appearance. Motif and floral designs are thought to have originated around the 18th century.

The traditional marbling process consists of three steps [1]. First of all, the background liquid is placed in a tray and the paints are sprinkled or dropped onto the liquid surface with eyedroppers or brushes to create an initial design. The liquid layer has to be thick enough to keep the paints floating on its surface. Then, styluses, combs (also called rakes), and other tools are used to change the initial pattern. The comb's tines are arranged with different spacings to create different effects. As the marbler runs the tools back and forth across the tray, the complex marbling design emerges. An intricate pattern usually requires several strokes. Once the pattern is completed, the marbler gently applies a sheet of paper, fabric, or some other material onto the tray in order to capture the pattern. The painting created on the surface is finally transferred to the contact material.

The patterning tools and how they are manipulated are crucial in producing impressive features. Many modern marblers are in search of new effects or techniques to heighten their expressivity. Unfortunately, it is not an easy task since the marbling process consists of several steps and one must start from scratch when any mistake is made.

Digital simulation systems based on complex physical models have been commonly used to create marbling images [2], [3]. However, these methods suffer from blurry contours because the time-iterative-relaxation nature of the solver makes dissipation inevitable. The more marbling operations are applied, the more blurry the result becomes. Therefore, it is difficult for them to produce publication-quality marbling images as fine features will be lost. This motivates us to find simple closed-form mathematical formulas to simulate the marbling process.

The major technical contribution of the paper is to provide simple yet closed-form deformation formulas for marbling process simulation, while avoiding the computational cost of full fluid simulation. It is an approximation that is rich enough to capture a large collection of phenomena. In addition to simplicity, the use of mathematical formulas is also advantageous in terms of control, speed, ease of implementation, parallelism, and vector output. It enables the generation of beautiful designs with real-time visual feedback and the progressive fluid-like illustration of the marbling

- *S. Lu and X. Jin are with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310027, P. R. China. E-mail: {lushufang, jin}@cad.zju.edu.cn.*
- *A. Jaffer is with TextMyFood LLC., One Broadway, 14th Floor Cambridge, Massachusetts 02142 USA. E-mail: agj@alum.mit.edu.*
- *H. Zhao is with Wenzhou University, Wenzhou, 325035, P. R. China. E-mail: hanlizhao@gmail.com.*
- *X. Mao is with the Department of Computer and Media Engineering, University of Yamanashi, 4-3-11 Takeda, Kofu, Yamanashi 400-0016, Japan. E-mail:mao@yamanashi.ac.jp.*

process.

The vector output of marbling images solves the dissipation problem completely, which is a challenging issue in existing physics-based methods, and preserves the sharp contours in the resulting images. We implement our design process on the programmable graphics hardware (GPU) to provide a real-time intuitive feedback. Moreover, various extensions and applications, including cyclic texture tiling, scene decoration, high-quality surface detail rendering on 3D objects, image editing, and interactive video processing, are implemented.

2 Related Work

As one of the traditional arts, marbling has been attracting attention from graphics researchers.

Early work on marbling simulation favored the use of numerical simulation. Physics-based simulations view the marbling process as a 2D computational fluid dynamics (CFD) problem and try to numerically solve complicated Navier-Stokes (NS) equations [4]. Akgun [5] developed a computer-aided paper marbling tool focused on generating the traditional Turkish art forms. In order to obtain the effect of fluid fluctuations at different levels, Acar and Boulanger [6] introduced a multiscale fluid model as well as a sharp fluid boundary method to simulate marbling. Although their method can model highly turbulent marbling effects, it does not deal with some traditional marbling patterns like comb patterns. Acar [2] proposed a level-set driven method which provided a flexible environment to model a wide range of flows and artistic effects in 2D, and applied it to marbling. All aforementioned methods were implemented on the CPU. These methods do not provide real-time feedback because they have to solve the time-consuming physics equations. Jin et al. [7] presented a novel digital marbling framework by solving Navier-Stokes equations on the GPU. But it suffered from blurry paint interfaces because of the dissipation. Xu et al. [3] improved the algorithm by complex high-order advection schemes to combat energy dissipation. The problem with this approach is that it may bring instability. Zhao et al. [8] employed an accurate yet fast third-order unsplit semi-Lagragian constrained interpolation profile method to reduce numerical dissipation while retaining stability.

Marbling images generated using CFD models suffer from three limitations. The first is dissipation. Researchers often apply complicated computational mechanisms to combat such an effect [9]. Dissipation can lead to blurring or mixing of colors, which do not conform to the sharp features of real-world marbling. Dissipation is inevitable in physics-based methods. Even higher order advection schemes like fifth-order B-Splines are unable to eliminate it completely [6]. The second is speed. Although physics-based simulations have motivated a number of papers proposing fast solvers for various scenarios, generating megapixel-sized images with real-time feedback is still a challenge. The third is control. Prior methods usually have many physical parameters (e.g., viscosity of liquids, force of manipulation, etc.) whose effects on marbling are not obvious.

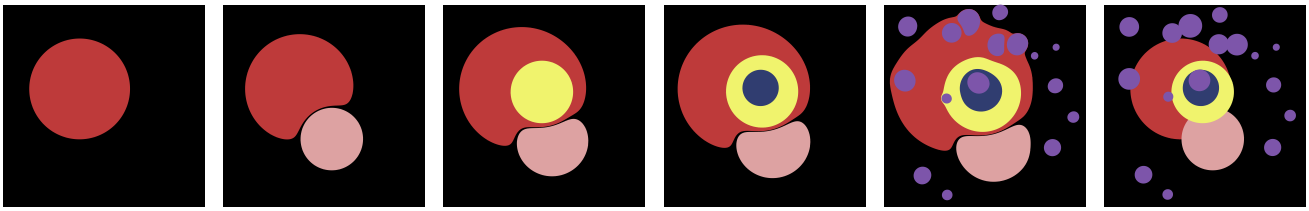
The commercial software Corel Painter [10] also supports a powerful image editing function which enables users to interactively create several traditional marbling effects. Corel Painter considers the marbling process as a distortion in which colors are dragged by comb and mixed into each other. These marbling effects are quite limited; only wavy path combing in horizontal and vertical directions is supported. Swirling patterns cannot be produced with this software. Moreover, their process is static and users do not receive any progressive feedback.

Though fluid-like turbulent motions can be simulated by numerically solving the fluid equations, it is desired to find a closed-form solution. Perlin [11] introduced a procedural turbulence noise function called Perlin noise to generate various kinds of 2D and 3D textures. Due to its efficiency and simplicity, it is frequently used to simulate fluid-like animations instead of physics-based methods. However, since Perlin noise functions have non-zero divergence, they cannot simulate incompressible fluids. Sims [12] used linear superposition of flow primitives to modulate velocity vector fields and warp 2D images to create the visual effect of flow. But neither of these methods is tailored for progressive, interactive marbling.

Vector graphics, in which primitives are defined geometrically, is an alternative representation of images. High-quality images can be obtained since the vector graphics representation is resolution-independent [13]. Vector images are used for rendering high-quality surface details on 3D objects even at high zoom levels [14], [15]. Wang et al. [16] introduced an effective vector representation for solid textures and mapped them onto mesh surfaces in real time. Ando and Tsuruno [17] presented a framework for generating marbled images that can be exported into a vector graphics format and allowed an arbitrary mouse-controlled interactive motion of a tine. However, the underlying fluid motion of this method requires numerically solving complicated and time-consuming physics equations. In this paper, we focus on the more intuitive mathematical marbling while supporting the output of vector geometries and high-quality surface details for rendering on 3D objects.

3 Our Approach

This paper presents a real-time digital system which enables artists to create marbling designs similar to those created by physical means. Our mathematical treatment of marbling starts with the assumptions of incompressible and immiscible 2-dimensional fluid inks. In this section we present in detail our tool function formulas based on topological computer graphics [18], which generate marbling designs with sharp contours and vector marbling outputs.



(a) (b) (c) (d) (e) (f)

Fig. 1. Ink-drop pattern: (a-e) the process of dropping color spots with deformation, and (f) the result without deformation.

3.1 Tool Functions

Our system mainly supports five types of patterning tools, which are described in the following subsections with forward transform formulas. These tools are all frequently used in the traditional marbling process. It is not difficult to prove that these tool functions are incompressible. A key feature of these marbling transforms is that the displacement parallel to a line is dependent only on the perpendicular distance from the line. Because of this, the backward transform is simply the forward transform with its displacement negated. Therefore, the proposed tool functions can all be applied forward and in the inverse. Forward application is used to generate vector-based output; inverse application is used to generate pixel-based output. In both cases, the closed-form expressions preserve the quality and sharpness of the design. Though the free-hand curve interaction tool is not supported directly, its effect can be simulated similarly by combining existing tools (Figure 10e).

In our mathematical marbling system, patterning is realized by the deformation of the current pattern under the applied tool functions. During the design process, we use the pixel-based image warping implemented on the GPU to achieve real-time immersive feedback. Specifically, to determine the color at a point $\mathbf{P}(x, y)$ in current pattern, we use the backward image mapping method to trace the trajectory for its former position $\mathbf{P}'$ in the previous pattern, and then copy the color at point $\mathbf{P}'$ to point $\mathbf{P}$. For designs started from blank page, as opposed to starting from a raster image, users can output the vector graphics representation of marbling images at the end of marbling process with some delay. Specifically, a regular n-polygon is used to approximate a paint drop with a predefined color; then the drop is forward transformed to create a warped one by displacing the polygon points.

3.1.1 Ink Drop Function

After pouring the substrate into a tray, the first step in the marbling process is to add paint drops onto the liquid. When applying floating color drops onto the liquid, previously dropped spots are deformed by the subsequently dropped ones and their shapes can change. Also, a spot may be dropped right within an existing one and the existing spot will be forced to spread to the surroundings. It is quite difficult to simulate this application process faithfully in physics-based systems. But a mathematical treatment produces a simple, exact solution. The first paint drop in the tray forms a circular spot with area a. If a second drop with area b is put in the center of the first drop, the total covered area increases from a to $a + b$. Points at the center move from radius 0 to radius $\sqrt{b/\pi}$, and boundary points move from radius $\sqrt{a/\pi}$ to $\sqrt{(a+b)/\pi}$. Therefore, this transformation is incompressible in the sense that the area of ink regions other than the one being injected (dropped) does not change. Given a point $\mathbf{P}$ and a paint drop centered at $\mathbf{C}$ with radius r, if $|\mathbf{P} - \mathbf{C}| < r$, point $\mathbf{P}$ is within the drop and takes its color, otherwise $\mathbf{P}$ is displaced radially from $\mathbf{C}$:

$$\mathbf{P}' = \mathbf{C} + (\mathbf{P} - \mathbf{C})\sqrt{1 + \frac{r^2}{|\mathbf{P} - \mathbf{C}|^2}}. \quad (1)$$

According to the transform function given above, the last ink drop is rendered last and has a round outline. However, the second to last drop is distorted by the last drop. Working backward through the ink droppings, each will be distorted by all the subsequent drops as shown in Figure 1(a-e).

Sprays of small droplets cause little distortion to underlying inks, yet require as much computation as if they were larger. So we provide another function with no deformation. If the spot overlaps with an existing one, the overlapped part is overwritten with the newly dropped one (see Figure 1f).

3.1.2 Tine-Line Pattern Function

This function manipulates a marbling by running tine lines through it in any direction. Consider the pattern transformation induced by drawing a single tine straight from one side of the tank to the other. Points near a tine's trajectory are moved in the direction of the tine's motion. The amount of motion in the tine's direction is (roughly) inversely proportional to the point's distance from the tine's trajectory. There are many possibilities for this inversely proportional function. The function we choose parameterizes both the maximum shift and sharpness of the shift gradient. Because the point's motion is perpendicular to the shortest distance between the point and the line, this displacement and its inverse are equally easily to calculate. So each cohort of points at a distance

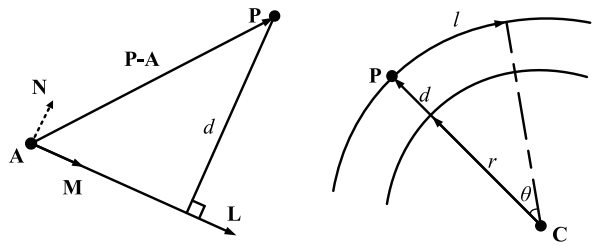
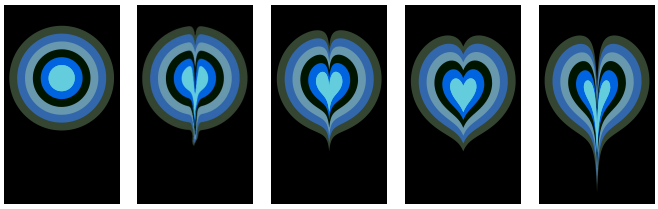


Fig. 2. Illustrations of tool functions: (a) the single tine-line and (b) the circular tine-line.



(a) original (b) Xu et al. (c) (80, 8) (d) (80, 32) (e) (240, 8)

Fig. 3. Tine-line pattern examples: (a) the original input, (b) the tine-line geometry motion made by Xu et al.'s system, and (c-e) our results of varying the value of (α,λ) with the same single tine-line operation.

d from the tine-line is shifted by the same amount. On an infinite plane, these parallel shifts neither compress nor expand the inks. In Figure 2a, if $\mathbf{L}$ is the tine-line with arbitrary slope, $\mathbf{N}$ is a unit vector perpendicular to $\mathbf{L}$, $\mathbf{A}$ is a point on the tine-line, and $\mathbf{M}$ is the unit vector in the direction of the tine-line $\mathbf{L}$, the mapping for point $\mathbf{P}$ is:

$$\mathbf{P}' = \mathbf{P} + \frac{\alpha\lambda}{d+\lambda}\mathbf{M}, \tag{2}$$

where $d = |(\mathbf{P} - \mathbf{A})\cdot\mathbf{N}|$ is the displacement function which represents the distance from point $\mathbf{P}$ to the tine line $\mathbf{L}$, and the scalars α and λ control the maximum shift and sharpness of the shift gradient (see Figure 3(c-e)). As the composition of homeomorphisms is a homeomorphism, the mapping of multiple parallel tine-line strokes can be composed into a single function. This composite function can comb in any direction (Figure 10a and 10b). A comb is used to create the classical patterns called Nonpareil (Figure 6d) and Gel Git (Figure 6e); it can be further modified to form many other traditional designs. For evenly spaced multiple tines which move as a rigid assembly, we define an alternative single function that represents the displacement from the set of parallel tines: $d' = s/2 - |\text{fmod}(d, s) - s/2|$, where d is the distance from $\mathbf{P}$ to an arbitrary tine-line of the comb and s is the spacing between tines. In this way, the mapping function is computed by replacing d in Eq. (2) with d'. Although the new function will bring discontinuous derivative because of the absolute value, it produces similar results with less computational cost. Figure 3b illustrates the same tine-line geometry and motion produced by Xu et al.'s system. In comparison, our tine-line pattern function is able to provide as visually plausible simulation effects as those made by physics-based methods.

3.1.3 Wavy Pattern Function

Curved tine trajectories contribute a lot to marbled images' beauty and charm. This function generates wavy patterns in any direction. In our model, the wavy path is a sinusoidal displacement (versus distance) applied after a (straight) comb operation described in Section 3.1.2. The result combined with two operations looks like the deformation from a comb stroked along a wavy path as shown in Figure 10(c, d, e). Let $f(v) = A\sin(\omega v + \psi)$ be the displacement function. By interactively specifying the amplitude A, the wavelength ω, and the phase ψ, users can move a comb in various wavy paths. With this operation, $\mathbf{P}$ is mapped to $\mathbf{P}'$ in the direction at angle t:

$$\mathbf{P}' = \mathbf{P} + f(\mathbf{P}\cdot[\sin t, -\cos t])[\cos t, \sin t]. \tag{3}$$

3.1.4 Circular Tine-Line Pattern Function

This function moves tines in circular trajectories controlled by the radius r and center $\mathbf{C}$ of the swirl. A circular tine-line motion is treated similarly to the tine-line motion. A circle is perpendicular to every radius ray at their intersection. For a given center point, motion along the arc containing a point $\mathbf{P}$ is made inversely proportional to the minimum radial distance from $\mathbf{P}$ to the tine circle. The concentric circles each being rotated by different amounts neither compress nor expand the inks. The scalars α and λ play the same role here as they do in the linear transformation of Section 3.1.2. In this case, as shown in Figure 2b, points are displaced along arcs around a center point $\mathbf{C}$. Under this operation, point $\mathbf{P}$ is mapped to:

$$\mathbf{P}' = \mathbf{C} + (\mathbf{P} - \mathbf{C})\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}, \tag{4}$$

where its angle subtended at $\mathbf{C}$ is $\theta = l/(|\mathbf{P} - \mathbf{C}|)$, the length of the displacement arc is $l = \alpha\lambda/(d+\lambda)$, and $d = ||\mathbf{P} - \mathbf{C}| - r|$.

It should be mentioned that the direction of the circular tine-line pattern depends on the sign of θ. That is, if θ is positive, it generates clockwise pattern, and vice versa. Figure 10f shows an example of the circular tine-line pattern.

3.1.5 Vortex Pattern Function

. Vortices, which wind more as the center is approached, are popular in marbling. The vortex pattern can be obtained using the same mapping function as Eq. (4) except the displacement term d. Different from the circular tine-line operation, we set the displacement $d = |\mathbf{P} - \mathbf{C}|$, where $\mathbf{C}$ denotes the center of the vortex (Figure 4).

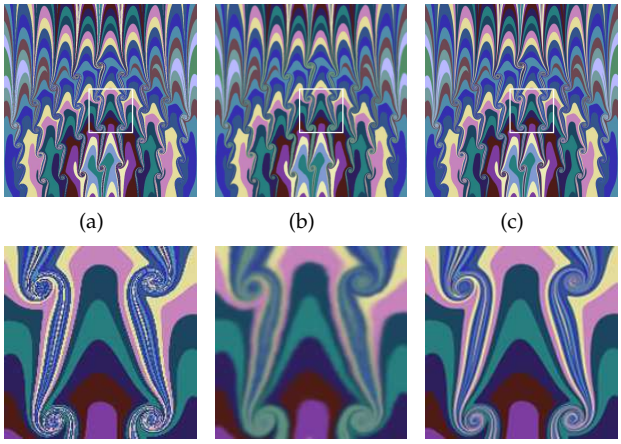


Fig. 4. Comparison of the image quality with different schemes: (a) aliasing artifacts, (b) blurry effects, (c) sharp contours with anti-aliasing, and (d-f) zoom in on images (a-c).

3.2 Preservation of Sharp Contours

Sharp contours between the paints are crucial characteristics of real-world marbling images. In this section, we describe how we employ several techniques to preserve sharp contours between the paints for a marbling of raster image inputs.

Since our method is based on the image deformation, it usually generates aliasing artifacts (see Figure 4a). The general strategy of antialiasing is to compute the image at a higher resolution and then down-sample it. In our implementation, we employ the 2×2 Rotated Grid Super-Sampling (RGSS) antialiasing scheme because of its low cost and high quality. Other common super-sampling patterns such as 4×4 grid, Quincunx, 4×4 checker, and 8-rooks are also provided [19].

In addition, if we update the image for each operation and use it as the input of the next operation, contrast fading occurs and the resulting image is blurred after some operations. Let n be the number of operations, the color at point $\mathbf{P}_n$ is:

$$C(\mathbf{P}_n) = C(\mathbf{P}_{n-1}) = \ldots = C(\mathbf{P}_1) = C(\mathbf{P}_0), \qquad (5)$$

where $\mathbf{P}_i (i = 0, 1, ..., n-1)$ is the back-traced point of $\mathbf{P}_{i+1}$. Signal diffusion is inevitable unless an ideal sinc filter is applied. Even high order interpolation methods like fifth-order B-Spline cannot prevent blurring completely [6]. As n increases, the result will get more and more blurry (see Figure 4b). To solve this problem, we employ an alternative image update scheme similar to Sims [12]. For each point in the current image, we trace its mapping point in the ink-drop pattern directly and copy the color at that point to the current position:

$$C(\mathbf{P}_n) = C(\mathbf{P}_n \rightarrow \mathbf{P}_{n-1} \rightarrow \ldots \rightarrow \mathbf{P}_1 \rightarrow \mathbf{P}_0). \qquad (6)$$

Consequently, the composition operation can be computed in one shader and the improved result is shown in Figure 4c. This scheme is also able to show the animation (evolution) of the marbling process, but unlike the first scheme, the performance of this scheme will decrease gradually as the number of operations increases. Therefore, our system makes both schemes available to users.

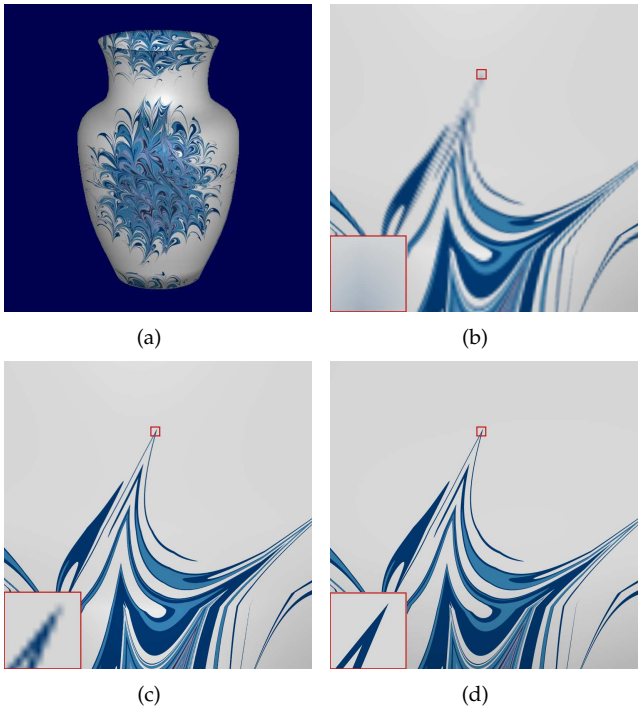


Fig. 5. Rendering high-quality surface details on 3D objects. (a) A vase rendered with a marbling pattern. (b) Traditional texture mapping with an 800×800 texture shows very blurry edges in a closeup view. (c) Even an 8000×8000 texture does not provide sharp edges for extreme closeups. (d) Our surface details rendering method guarantees crisp edges even for extreme closeups.

3.3 Vector Image Output

Our approach supports compact resolution-independent vector output when the initial pattern is created from paint drops. For each initial circular paint drop, we approximate it by an inscribed regular n-gon filled with the color of the drop. The value of n is chosen according to the radius of the drop. We transform the polygon points of the initial drop to new positions according to the composite formula derived from the marbling operations. The displacement of each point is very dependent on the marbling operations. Two adjacent points in an initial paint drop may be far apart after the marbling transformation; therefore under-sampling artifacts may occur. To reduce the artifacts, we employ an adaptive refinement scheme to keep the boundary smooth. As long as the distance between two transformed adjacent polygon points is larger than a user-specified threshold T, a new sampling point is inserted in the middle of the arc subtended by the two points in the initial paint drop. Empirically, pleasing results can be achieved if we set T

to be 1 pixel. A smaller threshold can be used for higher quality or when magnifying the marbling image.

4 Rendering Surface Details on 3D Objects

Given a parameterized 3D object, simply mapping the marbling image onto the 3D surface as a texture does not retain sharp texture features. Figure 5b is a case in point. Even a high-resolution texture (8000×8000) cannot faithfully reproduce the crisp boundaries for closeup views, as shown in Figure 5c.

Our closed-form solution makes it possible to render high-quality surface details on 3D objects ([14], [15]). After a marbling texture is designed, we record its creation history (including parameters for paint drops and operations). Then, we compute the marbling texture for 3D objects on-the-fly using the following two steps.
1. Render the 3D object using deferred shading technique to store per-pixel attributes (positions, normals and texture coordinates) into local video memory (G-buffer).
2. Render screen-aligned quadrilaterals by retrieving the per-pixel attributes stored in the G-buffer. To calculate the color of a pixel with its position located at $\boldsymbol{x}$, the pixel shader needs to access the following data structures. They are the buffer T used to store texture coordinates in the step 1, the patterning operations f, the paint drops $(\boldsymbol{col}, r, \boldsymbol{c})$ (These parameters respectively represent color, radius, and center.), the background color $\boldsymbol{bc}$, width w and height h of the marbling image. We present the pseudo code in Algorithm 1.

Algorithm 1: Vector Marbling Texture Fetch

```
tc ← T(x)
p ← tc * (w, h)
for all operations(f) do
  p ← f(p)
end for
for all paintdrops(col, r, c) do
  if ||p − c|| ≤ r then
    return col
  end if
end for
return bc
```

We implement the entire rendering process on the GPU to obtain the high efficiency. Such a method keeps sharp curves at high zoom levels in interactive frame rates and overcomes the limited resolution problem while reducing texture memory usage. A similar antialiasing scheme as in Section 3.2 is employed for the anti-alias pass. The resulting image is shown in Figure 5d.

5 Comparisons and Applications

We have implemented our system using HLSL shaders on the GPU. In order to investigate the performance of our algorithm, we have run the system on a 1.83GHz Intel Core 2 Duo E6320 CPU and an NVIDIA GeForce 8800GTS GPU. We show the comparison of performance of our method and Xu et al.'s approach in Table 1. It shows that our method with the 2×2 RGSS filter outperforms Xu et al.'s system by an order of magnitude. In comparison, the interactivity of Xu et al.'s system is limited to low resolutions due to the time-consuming physics simulation. Physics-based systems, including Acar's CPU-implemented level set method, cannot achieve real-time performance when designing high-resolution marbling images. We have also tested our system on a ThinkPad X61 laptop with an integrated Intel GMA X3100 graphics card and observed that the system can still achieve 13 FPS when designing 1 megapixel images using the 2×2 RGSS filter. As a result, not only can our method let users design their marbling images with real-time visual feedback, it can also enable them to experience the marbling process immersively because of its vivid fluid-like animations. Besides of its faster execution, our mathematical model is able to produce vector images (Figure 3c), whereas the pattern made by Xu et al.'s (Figure 3b) is a raster one.

Figure 6 shows the step by step procedure of generating a digital marbling. After the ink-drop pattern is made (Figures 6a through 6c), we draw a comb along a straight line from right to left (Figure 6d), and then draw back the same comb in the opposite direction with the teeth passing in between where they have passed before (Figure 6e). These two steps are repeated in the orthogonal direction (Figure 6f and 6g). Then, we perform two horizontal combings in opposite horizontal directions (Figure 6h and 6i). Finally, a wavy pattern function is applied (Figure 6j). Throughout this process, users can choose one of the schemes described in Section 3.2. The first one is fast but the very fine details that arise from repeated raking are blurred by this display process. However, the second scheme slows down as the number of operations increases, but the underlying model preserves all of the fine details without blurry artifacts. The resolution-independent vector images can be created at an intermediate stage, with some delay, and users can zoom in on particular regions to see the fine detail at any scale. (Figure 1 and Figure 3 show two simple results, and Figure 8b shows a complex one).

Figure 7 shows the comparison with the algorithm of Acar's level set method [2]. Compared with the real marbling pattern in Figure 7a, we can find that our mathematical marbling system can generate a pattern (see Figure 7c) visually similar to the result of physics simulation (see Figure 7b) under the same manipulation steps. Figure 8 is another example. It shows that our system can produce marbling effects comparable to real artwork.

The software Corel painter supports limited patterning tool functions. Most of the marbling patterns made by it are based on the ink-drop pattern function and tine-line pattern function in vertical and horizontal directions.

Resolution (in pixel)	Xu et al.'s approach	Ours			
		No-antialiasing	2×2 RGSS	4×4 checker	4×4 grid
512×512	30	1560	732	250	239
1024×512	15	1265	375	125	120
1024×768	10	850	246	83	80
1024×1024	12	660	189	64	61
1280×1024	7	548	156	55	53

TABLE 1
The performance comparison (in FPS) of our method and Xu et al.'s approach under the same environment.

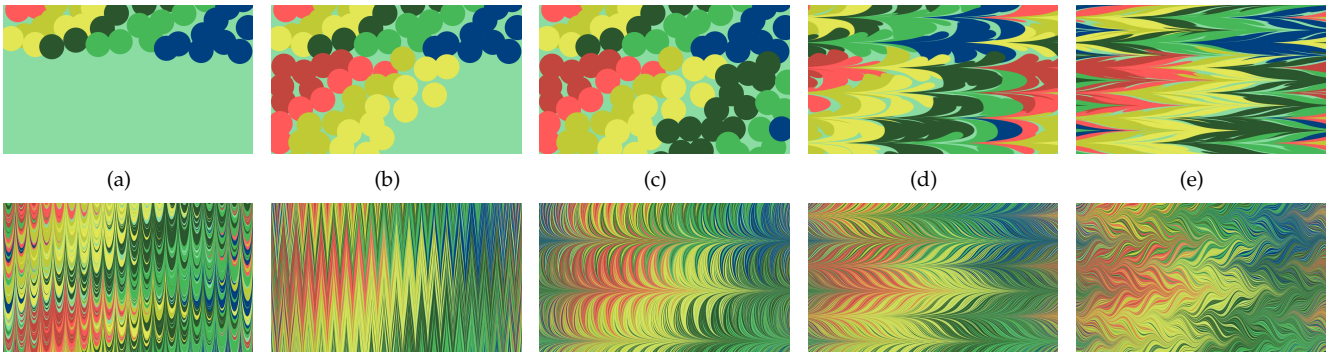


Fig. 6. Images showing the procedure of generating a digital marbling pattern step by step.

Fig. 7. Comparison of marbling images generated by different systems: (a) a real marbling pattern [Maurer-Mathison 1999], (b) result obtained from Acar's level set method, and (c) result obtained by our mathematical marbling system.

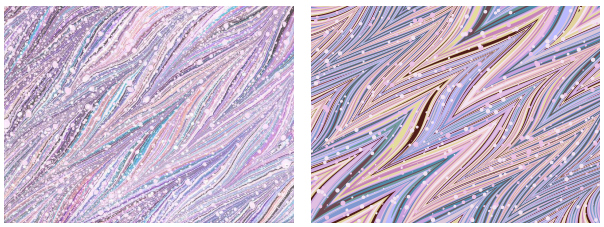


Fig. 8. Comparison between a real-world marbling pattern and our simulation result: (a) a real-world marbling image, (b) our simulation result. Zoom in with your PDF viewer to see how our vector marbling pattern preserves the very high levels of detail.

These two functions in our method are periodic as our system forces the part beyond the border of the image to enter the image again from the opposite side. Therefore, given a tileable input, the image produced by applying these two functions is still seamless whereas the one generated by Corel Painter has visible artifacts at seams.

We have applied our method to the field of image editing and video stream processing. Thanks to the high processing speed, it is possible to process video streams in real time. Actually, every frame of the video is a marbling pattern, and the interesting transformation of related video pictures can be potentially used in the area of film-making and advertisement as shown in the supplementary video.

We have also applied the images created with our system to scene decoration. The use of marbling textures for decorating a vase, a book and a tablecloth are shown in Figure 9a, a window in Figure 9b, and wallpaper, upholstery and carpeting in Figure 9c.

6 Conclusions

We have presented a novel method for modeling marbling processes using simple closed-form expressions. The method can generate authentic-looking marbling images, both vector and raster. Our real-time design process running on stock hardware is engaging and easy to learn and use. Amateurs can produce beautiful designs in a few minutes. Artists can use the system to try different designs before they create works with real marbling materials. For designs starting from a blank sheet, our approach can create resolution-independent vector-graphic images.

Real-world marbling processes involve complex fluid behaviors and chemical and physical interactions. We

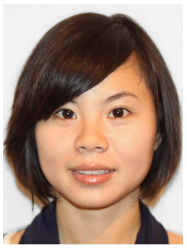
have been able to model a substantial variety of marbling effects with our tool functions, while avoiding the computational cost of full fluid simulation. We will explore more mathematical functions like free-hand tools to achieve a wider variety of effects in the near future. For the vector graphics output, the number of vertices grows quickly with the number of operations. This issue could be addressed by taking shape simplification into account. Alternatively, the current polygonal approximation to a deformed paint drop boundary curve could be subdivided at the instant that a tine crosses it. We also plan to design marbling solid textures by extending our focus to three dimensions, inspired by the research of [16].

Acknowledgments

The authors would like to thank Diane Maurer-Mathison (http://www.dianemaurer.com/) and Galen Berry (http://marbleart.us/AntiqueSpot.htm) for their permissions to use the marbling textures in Figure 7a and Figure 8a respectively. Xiaogang Jin was supported by the China 973 program (Grant no. 2009CB320801), the NSFC-MSRA Joint Funding (Grant no. 60970159), Zhejiang Provincial Natural Science Foundation of China (Grant no. Z1110154) and the National Natural Science Foundation of China (Grants 60933007 and 60833007). Hanli Zhao was supported by the Zhejiang Provincial Natural Science Foundation of China (Grant No. Y1110004).

References

[1] D. Maurer-Mathison, *The Ultimate Marbling Handbook: A Guide to Basic and Advanced Techniques for Marbling Paper and Fabric*. Watson-Guptill Publishing, 1999.
[2] R. Acar, "Level set driven flows," *ACM Trans. Graph.*, vol. 26, no. 4, p. 15, 2007.
[3] J. Xu, X. Mao, and X. Jin, "Nondissipative marbling," *IEEE Computer Graphics and Applications*, vol. 28, no. 2, pp. 35–43, 2008.
[4] J. Stam, "Stable fluids," in *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1999, pp. 121–128.
[5] B. Akgun, "The digital art of marbled paper," *Leonardo*, vol. 37, no. 1, pp. 49–52, 2004.
[6] R. Acar and P. Boulanger, "Digital marbling: a multiscale fluid model," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 600–614, 2006.
[7] X. Jin, S. Chen, and X. Mao, "Computer-generated marbling textures: a GPU-based design system," *IEEE Computer Graphics and Applications*, vol. 27, no. 2, pp. 78–84, 2007.
[8] H. Zhao, X. Jin, S. Lu, X. Mao, and J. Shen, "AtelierM++: a fast and accurate marbling system," *Multimedia Tools and Applications*, vol. 44, no. 2, pp. 187–203, 2009.
[9] B. Kim, Y. Liu, I. Llammas, and J. Rossignac, "Advections with significantly reduced dissipation of diffusion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 135–144, 2007.
[10] R. Grossman, *Digital Painting Fundamentals with Corel Painter 11*. Course Technology PTR, 2009.
[11] K. Perlin, "An image synthesizer," *Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.
[12] K. Sims, "Choreographed image flow," *The Journal of Visualization and Computer Animation*, vol. 3, no. 1, pp. 31–43, 1992.
[13] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin, "Diffusion curves: a vector representation for smooth-shaded images," *ACM Trans. Graph. (SIGGRAPH 2008)*, vol. 27, no. 3, p. 92, 2008.
[14] D. Nehab and H. Hoppe, "Random-access rendering of general vector graphics," *ACM Trans. Graph. (SIGGRAPH Asia 2008)*, vol. 27, no. 5, p. 135, 2008.
[15] S. Jeschke, D. Cline, and P. Wonka, "Rendering surface details with diffusion curves," *ACM Trans. Graph. (SIGGRAPH Asia 2009)*, vol. 28, no. 5, p. 117, 2009.
[16] L. Wang, K. Zhou, Y. Yu, and B. Guo, "Vector solid textures," *ACM Trans. Graph. (SIGGRAPH 2010)*, vol. 29, no. 4, p. 86, 2010.
[17] R. Ando and R. Tsuruno, "Vector graphics depicting marbling flow," *Computers and Graphics*, vol. 35, no. 2, 2011.
[18] A. Jaffer, "Topological Computer Graphics," http://people.csail.mit.edu/jaffer/Marbling, 2007.
[19] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-Time Rendering 3rd Edition*. Natick, MA, USA: A. K. Peters, Ltd., 2008.



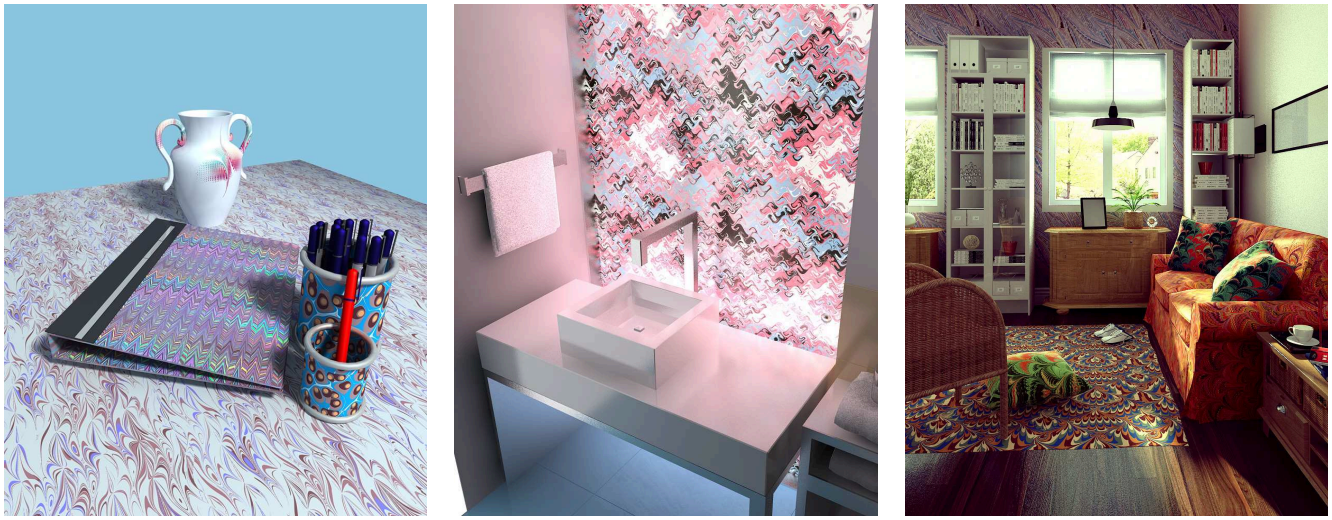
Shufang Lu is a PhD candidate at the State Key Laboratory of CAD & CG at Zhejiang University. She received her BSc degree in software engineering from Wuhan University in 2007. Her research interests include marbling simulation, non-photorealistic rendering and image processing. Contact her at lushufang@cad.zju.edu.cn.

Aubrey Jaffer is a mathematician and one of the founders of TextMyFood LLC. 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. Contact him at agj@alum.mit.edu.

Xiaogang Jin is a professor at the State Key Laboratory of CAD & CG at Zhejiang University. His research interests include implicit surface computing, cloth animation, crowd and group animation, texture synthesis, and digital geometry processing. Jin received his PhD in applied mathematics from Zhejiang University. Contact him at jin@cad.zju.edu.cn.

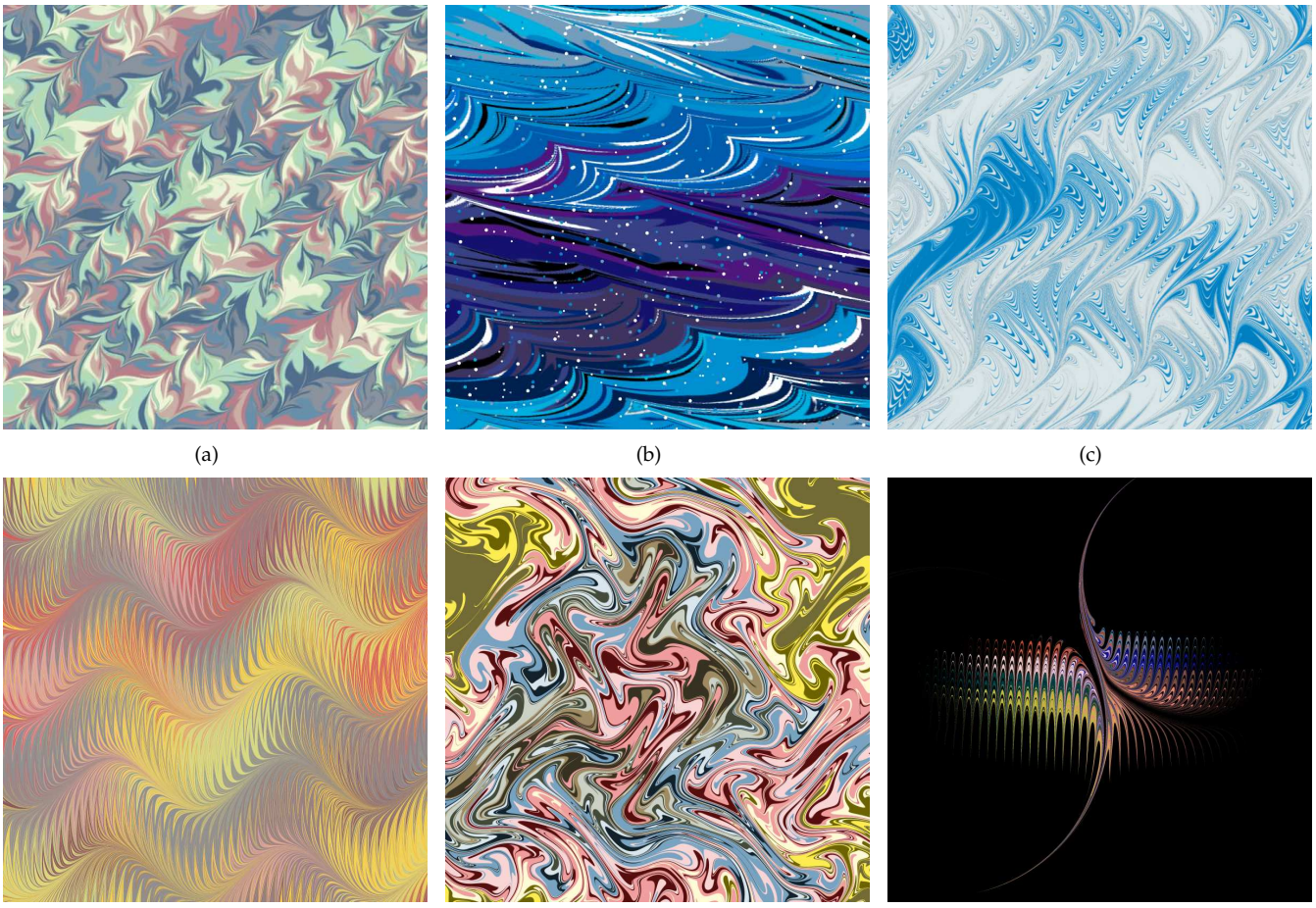
Hanli Zhao is a faculty member of Intelligent Information Systems Institute, Wenzhou University, Wenzhou, China. He received his B.Eng. degree in software engineering from Sichuan University in 2004 and his Ph.D. degree in computer science from Zhejiang University in 2009. His research interests include non-photorealistic rendering and general purpose GPU computing. Contact him at hanlizhao@gmail.com.

Xiaoyang Mao is a professor at the University of Yamanashi in Japan. Her research interests include non-photorealistic rendering, texture synthesis, perception and affect based rendering and visualization. Xiaoyang Mao recieved her BS in computer science from Fudan University in China and her MS and PhD in computer science from the University of Tokyo. Contact her at mao@yamanashi.ac.jp.



(a) (b) (c)

Fig. 9. Application to scene decorations: (a) the vase, book and tablecloth decorations, (b) window design, and (c) wallpaper, upholstery and carpeting in room decorations. These 3D scenes were rendered with ray tracing in 3D Studio Max.



(a) (b) (c)

(d) (e) (f)

Fig. 10. Various marbling images made by our mathematical marbling system.